Distribuidor en exclusiva
para España:



Valencia, 49-53 - 08015 Barcelona
Ardemans, 8 - 28028 Madrid

commodore

COMMODORE 64

MANUAL DEL USUARIO

Commodore Business Machines, Inc.
y
Howard W. Sams & Co., Inc.

D.L.B. 18979-83



INDICE

microelectronica
y control s a
MEC


microelectronica y control s a


microelectronica y control s a

Estamos conectando el **COMMODORE 64** a otras partes del equipo... Su sistema puede ampliarse añadiendo accesorios, conocidos como periféricos, si su sistema necesita crecer. Algunas de sus opciones incluye aparatos como un grabador DATASSETTE* o hasta 4 unidades de almacenamiento de disco VIC-1541 para guardar programas y/o datos. Puede añadir una impresora de matriz de agujas del VIC para disponer de copias impresas de sus programas, cartas, facturas, etc... Finalmente existe una de las posibilidades más interesantes por la gran variedad de aplicaciones de software disponible en CP/M**, al **COMMODORE 64** se puede conectar un cartucho que contiene el microprocesador Z-80.

Tan importante como todo el hardware disponible es el hecho de que el **MANUAL DE USUARIO** le ayudará a desarrollar sus conocimientos de ordenadores. No le explicará todo aquello que es preciso saber sobre ordenadores, pero se referirá a una gran variedad de publicaciones para información más *detallada sobre* los *temas presentados*. Commodore quiere que disfrute realmente de su nuevo **COMMODORE 64**. Y divirtiéndose, recuerde: aprender a programar no es cosa de un día. Sea paciente consigo mismo y siga el MANUAL DEL USUARIO. Bienvenido a todo un nuevo mundo de diversión!!

NOTA:

Muchos programas estaban en desarrollo a la hora de producir este manual. Por favor entre en contacto con su distribuidor local Commodore y con la revista Club Commodore y Clubs de Usuarios, los cuales le pondrán al día de la existencia de programas de aplicaciones, escritos mundialmente para el Commodore 64.

---

*DATASSETTE es una marca registrada por Commodore Business Machines, Inc.

**CP/M es una marca registrada por Digital Research Inc. Especificaciones sujetas a cambio.



DESEMBALADO Y CONEXION DEL COMMODORE 64

Las siguientes instrucciones paso a paso le enseñan como conectar el Commodore 64 a su receptor de televisión, sistema de sonido o monitor y como estar seguro de que todo está trabajando correctamente.

Antes de conectar algo al ordenador, compruebe el contenido de la caja del Commodore 64. Además de este manual, debería encontrar los siguientes artículos:

1. Commodore 64
2. Fuente de Alimentación (caja marrón con los enchufes de red y alimentación)
3. Cable Video

Si alguno de estos artículos faltara póngase en contacto con su distribuidor para recibirlos.

Primero, veamos la distribución de conexiones del ordenador y sus funciones.

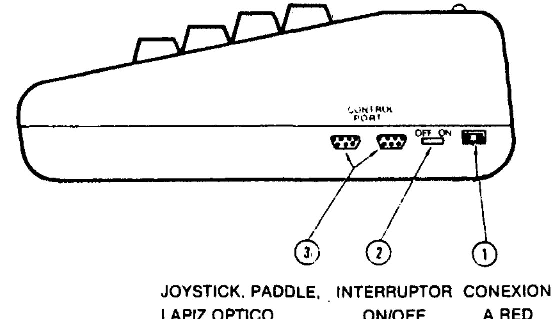
CONEXIONES DEL PANEL LATERAL

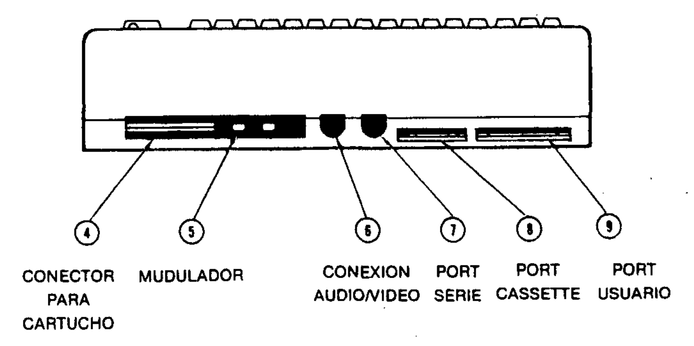
1. Conexión de Alimentación. El extremo libre del cable de la fuente de alimentación se conecta a la toma de alimentación del Commodore 64.
2. Interruptor de Alimentación. Pone en marcha al Commodore 64.
3. Ports de Juegos. Cada conector de juegos puede aceptar un joystick o control de juego paddle, mientras el lápiz óptico solo puede ser conectado en el port de juego cercano al frontal de su ordenador.

CONEXIONES TRASERAS

4. **Conexión para Cartuchos.** La conexión rectangular de la izquierda acepta programas o cartuchos de juegos.
5. **Conector TV.** Este conector transmite imagen y sonido a su receptor de televisión.
6. **Salida de Audio y Video.** Este conector suministra directamente señal de audio, que se puede conectar a un sistema de sonido de alta calidad, y una señal de video "compuesta", por si se dispone de un monitor de TV.
7. **Port Serie.** Puede conectar una impresora o unidad de discos directamente al Commodore 64 por medio de este conector.







8. **Interface Cassette.** Un grabador DATASSETTE puede conectarse al ordenador para grabar información, programas y/o datos.
9. **Port de Usuario.** Varios cartuchos de interface puede conectarse en el port de usuario, como el cartucho de comunicaciones RS-232.

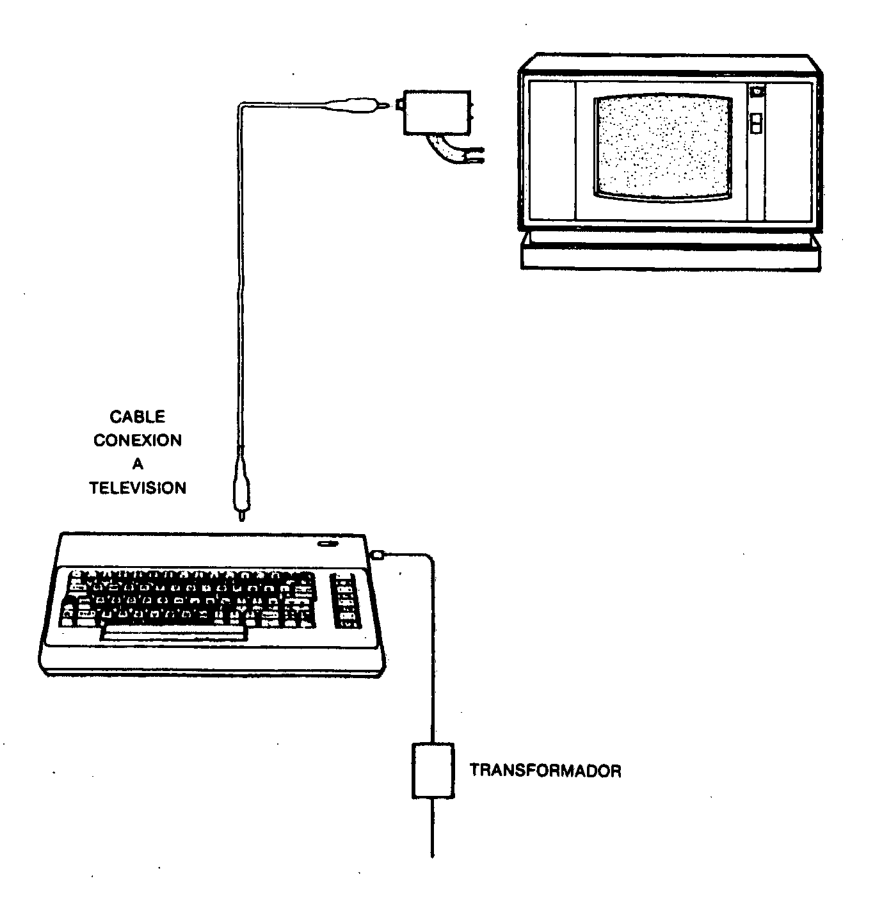
INSTALACION

CONEXIONES A SU TELEVISOR

El dibujo de las conexiones del ordenador al TV se muestra en la página 4.

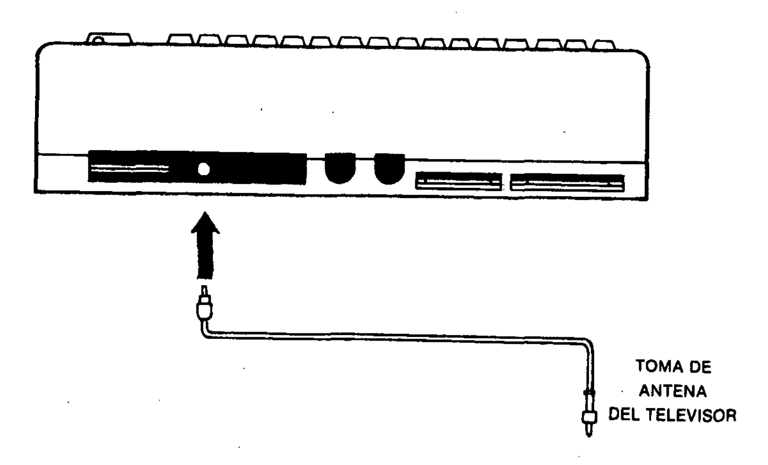
1. Conecte un extremo del cable de la TV al jack de señal tipo fono detrás del Commodore 64. Asegúrese de que está bien conectado.
2. Conecte el otro extremo del cable al conector de antena de su TV.
3. Si tiene antena de UHF, desconéctela de su receptor de TV.
4. Enchufe el cable de red a la toma de corriente eléctrica.





5. Enchufe el cable de la fuente de alimentación en el conector de alimentación en el lateral del Commodore 64. Es necesario tener en cuenta que la inserción solo se efectúa en una sola posición, así pués no se puede conectar el cable de la alimentación de una manera incorrecta. La fuente de alimentación convierte la corriente doméstica en la forma que utiliza el ordenador.

El Commodore 64 ahora está correctamente conectado. No se requieren conexiones adicionales para el uso del ordenador con su TV.



CONEXIONES OPCIONALES

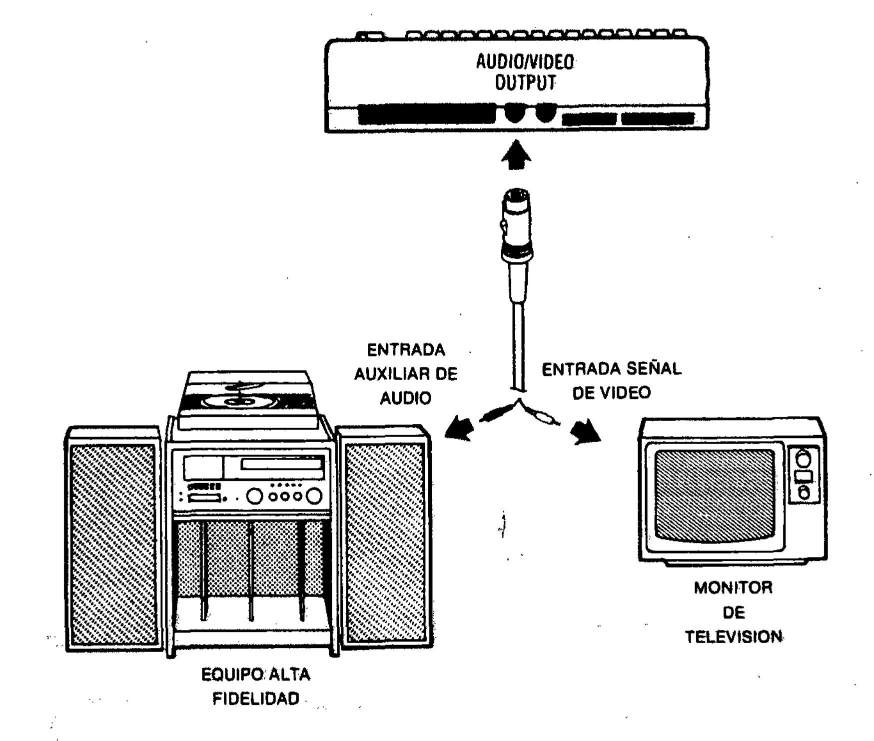
Como el Commodore 64 está equipado con un canal de sonido en Alta Fidelidad, puede reproducirse a través de un amplificador de calidad para obtener el mejor sonido posible. Además, el Commodore 64 también está provisto de una señal de video "compuesta" standard, la cual puede alimentar un monitor de televisión.

Estas opciones son posibles por el jack de salida audio/video en el panel trasero del Commodore 64. La sencilla vía para acceder a estas señales es utilizando un cable de audio DIN standard de 5 patillas (no se suministra). Este cable se conecta directamente al conector de audio/video en el ordenador. Dos de las cuatro patas en el extremo opuesto del cable contienen las señales de audio y video. Opcionalmente, puede construirse su propio cable, usando como guía la muestra de las patas de salida en el Apéndice I.

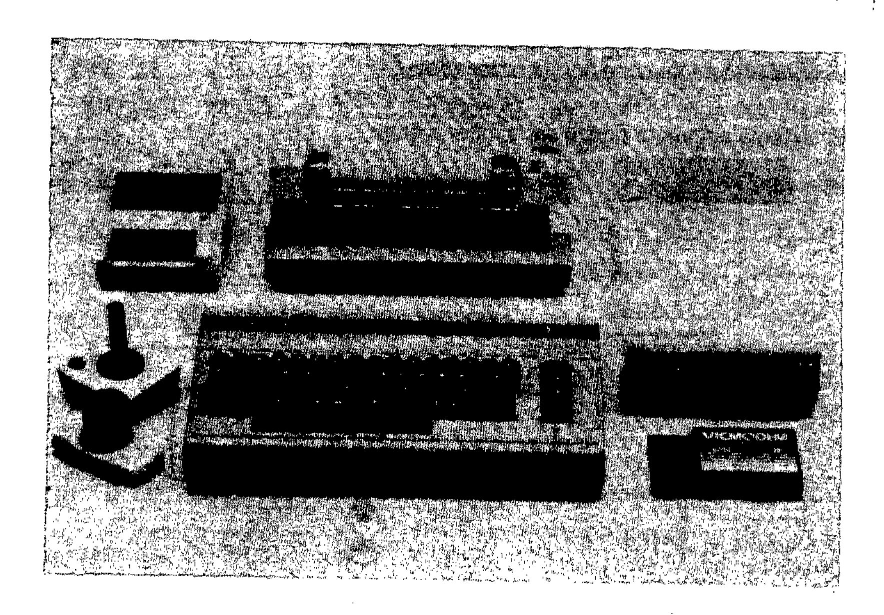


Si compra equipos periféricos, tal como una unidad de discos VIC-1541 o una impresora VIC-1525, puede necesitar conectarlos al mismo tiempo. Consulte los manuales de usuario entregados con el equipo adicional para proceder correctamente en su conexión al ordenador.

El sistema completo puede ser similar a este:







USO DEL COMMODORE 64

1. Ponga en marcha el ordenador utilizando el interruptor bascular en el panel del lado derecho si miramos el ordenador de frente.
2. Después de unos momentos se verá lo siguiente en la pantalla del televisor:





3. Si su TV tiene botón de sintonía fina manual, ajuste el TV hasta que tenga una imagen clara.
4. Puede también necesita ajustar los controles de color y de brillo/contraste en el TV para obtener la mejor imagen. Puede utilizarse el procedimiento de ajuste de color descrito más adelante para obtener el tono correcto de color. Cuando vea la primera imagen, la pantalla aparecerá en su mayor parte azul oscuro, con borde y letras azul claro.

Si no obtuviera los resultados requeridos, repase los cables y conexiones. La tabla adjunta le ayudará a aislar cualquier problema.

Síntoma	Causa	Solución
La luz indicadora está apagada	El interruptor no está encendido.	Asegúrese que el interruptor esta en la posición "ON"
	El cable de red no está conectado.	Revise si el cable está en su sitio.
	Fusible fundido	Lleve el equipo a su distribuidor para que se le revise
No sale la imagen	TV. en otro canal	Busque la señal con el mando de sintonía del TV
	Conexión incorrecta	Revise la conexión del ordenador al TV.

Síntoma	Causa	Solución
Caracteres aleatorios en la pantalla	Cartucho mal conectado	Reinserte el cartucho
Imagen sin color	TV fuera de sintonia	Resintonice el TV
Imagen pobre de color	Ajuste incorrecto del color y brillo/contraste	Reajuste los mandos de color y brillo/contraste
Excesivo ruido de fondo	Volumen demasiado alto	Ajuste el mando de volumen
Imagen correcta pero sin sonido	Volumen demasiado bajo	Ajuste el mando de volumen

AVISO: El **COMMODORE 64** ha sido diseñado para ser utilizado por cualquier persona.
Pero en COMMODORE sabemos que pueden existir usuarios que encuentren dificultades. Para ayudar a responder a sus preguntas y adquirir algunas interesantes ideas de programación, hemos creado varias publicaciones para ayudarle. Puede también buscar ayuda uniéndose a un Club de Usuarios de Ordenadores Personales de COMMODORE para compartir sus experiencias.

CURSOR

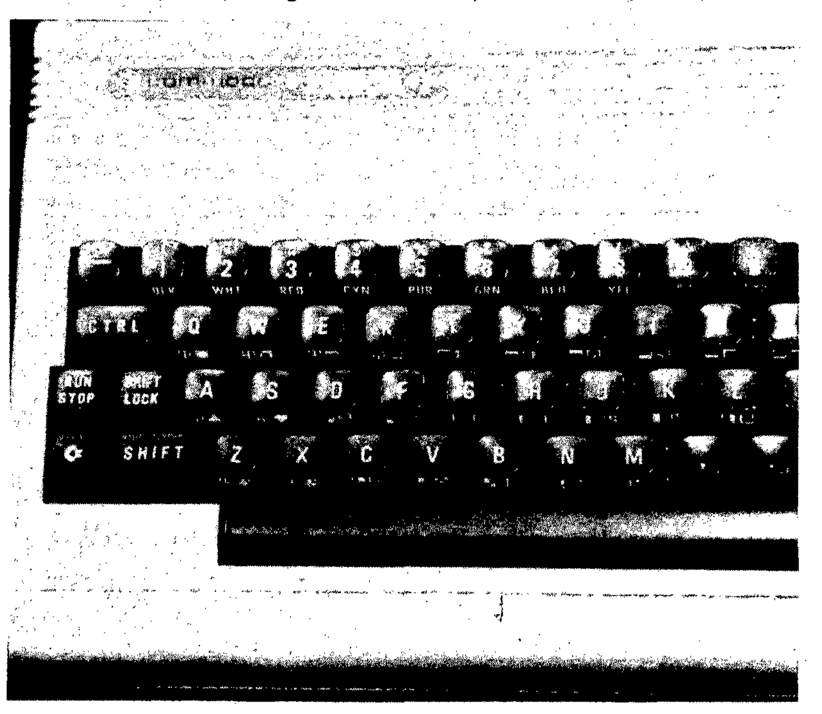
El cuadrado parpadeante que aparece debajo de READY se llama cursor e indica que lo que se pulsa en el teclado sera exhibido en la pantalla. Cuando pulse alguna tecla, el cursor se moverá delante un espacio, y la posición original del cursor será reemplazada por el caracter que ha pulsado. Pruebe a pulsar en el teclado y vea como los caracteres que ha pulsado aparecen en la pantalla del televisor.



AJUSTE DE COLOR

Hay una manera simple de tener un patrón de colores en el televisor, asi puede ajustar fácilmente el receptor. Aunque puede no estar familiarizado aún con el ordenador, siga adelante, y verá que fácil es el uso del Commodore 64.

Primero, mire al lado izquierdo del teclado y localice la tecla marcada CTRL . Es abreviatura de ConTRoL y se usa en unión de otras teclas, para ordenar a la máquina que haga unas tareas especificas.

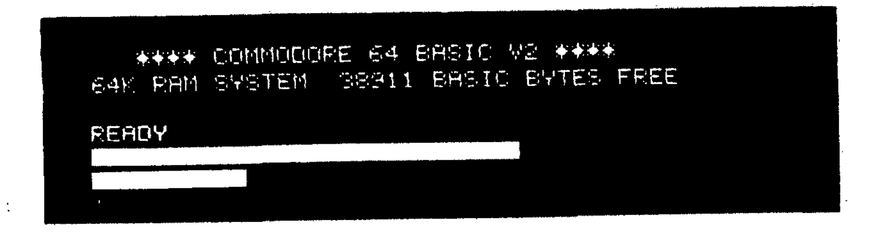


Para utilizar una función de control, presione la tecla CTRL y manténgala asi mientras pulsa una segunda tecla.

Pruebe lo siguiente: presione la tecla CTRL mientras también presiona la tecla 9. Entonces libere ambas teclas. Nada habrá sucedido, pero si pulsa alguna tecla ahora, la pantalla presentará el caracter visto en modo inverso (inversión de color), en lugar del modo normal.

Pulse la BARRA ESPACIADORA . ¿Qué pasa? Si ha realizado lo anterior correctamente, verá una barra azul claro moviéndose a través de la pantalla que crecerá hasta la siguiente línea si mantiene la presión en BARRA ESPACIADORA.



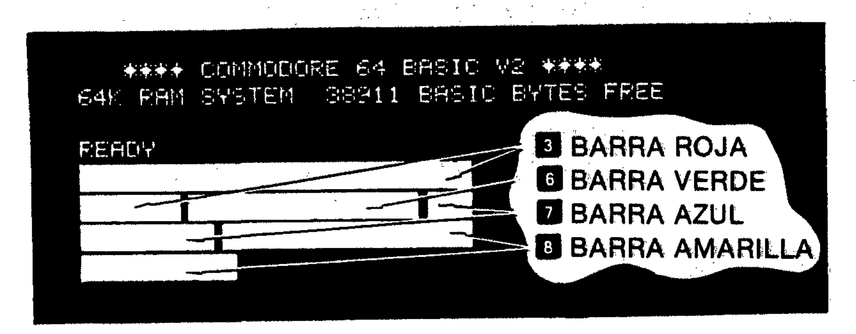


Ahora, pulse CTRL mientras presiona alguna de la otras teclas numéricas. Cada una de ellas tiene el nombre de un color (en inglés) marcado delante. Pulsando la tecla correspondiente y CTRL se obtiene el color. Por ejemplo, pulse CTRL y la tecla 8 y libere ambas. Ahora pulse ESPACIO BARRA.

Observe la pantalla: La barra ahora es amarilla. De manera similar puede cambiar la barra a cualquiera de los otros colores indicados en las teclas numéricas pulsando CTRL y la tecla apropiada.

Cambie la barra a diferentes colores y entonces ajuste los controles de color y matiz de su TV para obtener así una presentación a su gusto.

En la pantalla aparecerá algo similar a esto:



En este punto todo está debidamente ajustado y trabaja correctamente. Los siguientes capítulos le introducirán en la programación con el lenguaje BASIC. De cualquier modo, puede empezar inmediatamente usando algunas de las muchas aplicaciones ya escritas y juegos disponibles para el Commodore 64 sin conocer nada sobre programación de ordenadores.

Cada uno de estos paquetes contiene información detallada sobre como usar el programa. Le sugerimos, sin embargo, que lea completamente estos primeros capítulos de este manual para familiarizarse más con la operación básica de su nuevo sistema.



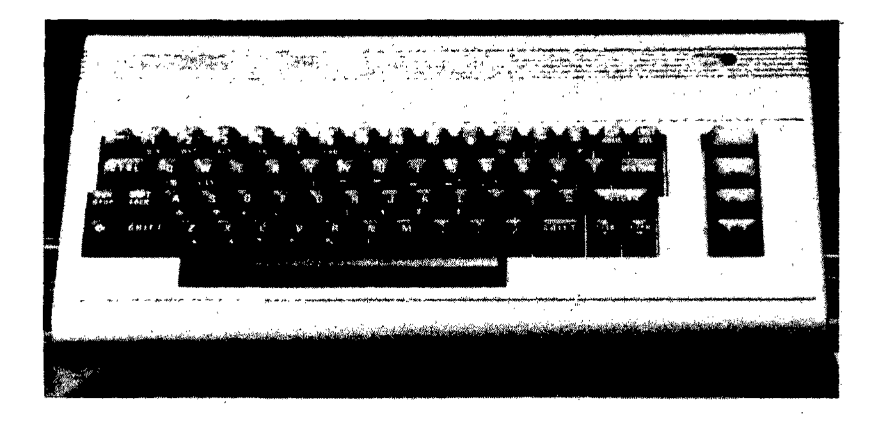
capítulo 2

EMPIECE A UTI[illegible] SU O[illegible]

EL TECLADO

Ahora que ha terminado de instalarlo y ajustarlo todo, por favor tómese unos momentos para familiarizarse con el teclado, que es el más importante medio de comunicación con su Commodore 64.

Encontrará el teclado similar al de una máquina de escribir standard en muchos aspectos. Hay, sin embargo, cierto número de nuevas teclas las cuales controlan funciones especializadas. Lo que sigue es una breve descripción de la teclas y cuál es su función. La operación detallada de cada tecla será incluida en secciones posteriores.



RETURN

La tecla RETURN le indica al ordenador que la información que se ha tecleado está completa y que puede procesar dicha información.

SHIFT

La tecla SHIFT trabaja de manera similar a la de una máquina de escribir estandard. Muchas de las teclas son capaces de presentar dos letras o símbolos y dos caracteres gráficos. En el "modo mayúsculas/minúsculas" la pulsación de la tecla SHIFT facilita los caracteres standar en mayúsculas. En el "modo mayúsculas/gráficos" la tecla SHIFT presentará el caracter gráfico del lado derecho de la parte frontal de la tecla.

En el caso de las cuatro teclas de función, la tecla SHIFT realiza la función marcada en el frontal de la tecla.



EDICION

Nadie es perfecto, y el Commodore 64 lo tiene en cuenta. Las teclas de edición permiten corregir errores de escritura y mover información por toda la pantalla.

CRSR

Hay dos teclas marcadas CRSR (de CuRSoR), una con flechas hacia arriba y abajo ⇕ CRSR ⇕ y la otra con flechas hacia derecha e izquierda ⇐CRSR⇒. Puede utilizar estas teclas para mover el cursor arriba y abajo o a izquierda y derecha. En el modo normal, las teclas CRSR permiten mover el cursor abajo y a la derecha. Pulsando al mismo tiempo la tecla SHIFT y las teclas CRSR el cursor se mueve arriba o a la izquierda. Las teclas de cursor tienen una característica especial de repetición para mantener el cursor en movimiento sin soltar la tecla.

INST/DEL

Si pulsa la tecla INST/DEL, el cursor se moverá un espacio a la izquierda, borrando (DELeting) el caracter anterior que se ha pulsado. Si está en medio de una línea el caracter de la izquierda se borra y los caracteres de la derecha automáticamente se mueven unidos cubriendo el espacio.

SHIFT e INST/DEL permite INSerTar información en una línea. Por ejemplo, si advierte que ha cometido un error al principio de una línea —quizás ha omitido una letra— puede usar la tecla ⇐CRSR⇒ para mover el cursor a un espacio después del error y entonces pulsar SHIFT e INST/DEL para insertar un espacio. A continuación pulse la letra que falta.

CLR/HOME

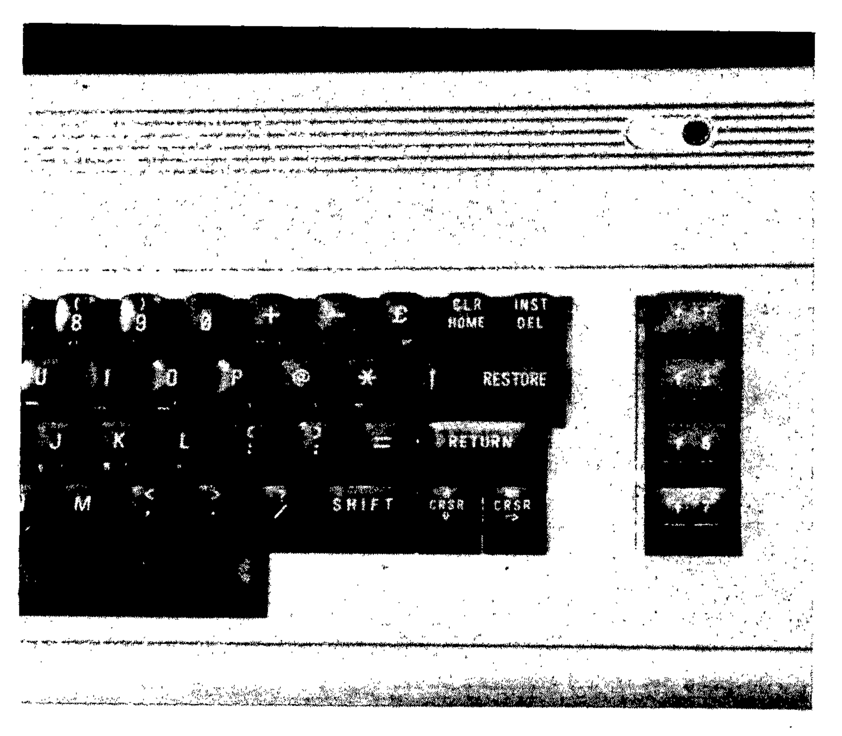
Esta tecla posiciona el cursor en la esquina superior izquierda de la pantalla. SHIFT y CLR/HOME borrará la pantalla y situará el cursor en la posición antes mencionada.

RESTORE

RESTORE opera como el nombre indica. Esto restablece el estado normal del ordenador como era antes de cambiar con un programa o algún comando. Una explicación más detallada se dará en capítulos posteriores.

TECLAS DE FUNCION

Las cuatro teclas de función en la parte derecha del teclado pueden ser "programadas" para realizar una gran variedad de funciones. Se pueden definir de muchas maneras para manejar tareas repetitivas.



CTRL

La tecla CTRL, cuyo nombre es una abreviatura de ConTRoL, permite escojer colores, y ejecutar otras funciones especializadas. Pulse la tecla CTRL mientras presiona otra tecla designada para acceder a una función de control. Ha tenido oportunidad de probar la tecla CTRL cuando cambiaba los colores del texto para crear diferentes barras de color durante el proceso de instalación.

RUN/STOP

Normalmente, pulsando la tecla RUN/STOP parará la ejecución de un programa BASIC. Utilizando la tecla RUN/STOP en el "modo SHIFT" permitirá la

carga automática de un programa desde la cinta y su ejecución sin tener que pulsar RUN.

C= LA TECLA COMMODORE

La tecla señalada con el logotipo de COMMODORE realiza cierto número de funciones: La primera permite moverse entre los modos de presentación gráfico y texto.

Cuando el ordenador se pone en marcha, está en modo Mayúscula/Gráfico, es decir, todo lo que escriba estará en letras mayúsculas. Como se ha mencionado, utilizando la tecla SHIFT en este modo se exhibirán los gráficos del lado derecho de las teclas.

Si pulsa las teclas C= y SHIFT, cambia la presentación de mayúsculas a minúsculas. Ahora bien, si pulsa la tecla C= y cualquier otra tecla con símbolo gráfico, se exhibe el gráfico del lado izquierdo de la tecla que se ha pulsado.

Volvemos al modo mayúscula/gráfico pulsando las teclas C= y SHIFT de nuevo.

La segunda función de la tecla C= es permitir el acceso al segundo juego de 8 colores de texto. Pulsando la tecla C= y alguna de las teclas numéricas, cualquier texto que ahora se escriba estará en el color alternativo disponible de la tecla que se pulsó. En el capítulo 5 encontrará una lista de los colores de texto asociados con cada tecla.

RETORNO A LA NORMALIDAD

Hasta aqui Vd. ha tenido oportunidad de ver suficientemente el teclado, ahora estudiaremos algunas de las muchas capacidades del Commodore 64.

Si aun tiene en pantalla las barras de color para el ajuste de su receptor de televisión, pulse SHIFT y CLR/HOME. La pantalla se limpiará y el cursor se situará en el punto "HOME" (esquina superior izquierda de la pantalla).

Ahora, pulse simultáneamente C= y la tecla 7. Esto selecciona el color de texto en azul claro. Hay otro paso necesario para volver a la normalidad. Pulse CTRL y 0 (el número "0", no la letra "O"). Esto situa la presentación en las condiciones normales. Si recuerda, se pone en modo INVERSO con CTRL 9 para crear las barras de color (las barras de color eran espacios inversos). Si estaba en el modo texto normal durante el test de colores, el cursor se habría movido, pero sólo espacios vacios a la izquierda.

UN INCISO:

Ahora que ha hecho las cosas por la vía difícil, podemos ver una manera simple de devolver a la máquina a su presentación normal. Primero pulse la tecla RUN/STOP y manteniéndola apretada pulse la tecla RESTORE . RUN/STOP debe siempre pulsarse antes que la tecla RESTORE .

Esto limpia la pantalla y vuelve todo a la normalidad. Si hay un programa en el ordenador, no será tocado. Esta es una buena secuencia a recordar, especialmente si practica la programación.

Si desea devolver la máquina a las condiciones anteriores como si hubiera apagado y encendido el ordenador de nuevo, escriba: SYS 64759 y pulse RETURN . Tenga cuidado al usar este comando. Borrará cualquier programa o información que contuviera el ordenador.

CARGA Y GRABACION DE PROGRAMAS

Una de las más importantes características del Commodore 64 es su capacidad de guardar y cargar programas en y desde cinta cassette o disco.

Esto permite guardar sus programas escritos para su uso posterior, o comprar programas pregrabados para utilizar con el Commodore 64.

Asegúrese que ambos, la unidad de disco o el DATASSETTE están conectados debidamente.

CARGA DE PROGRAMAS PREGRABADOS

Para todos los interesados en utilizar solamente programas pregrabados disponibles en cartuchos, cassette, o disco a continuación se describe todo lo que han de hacer:

1. **CARTUCHOS:** El ordenador Commodore 64 tiene una línea de programas y juegos en cartuchos. Los programas ofrecen una gran variedad de aplicaciones de negocios y personales y los juegos son similares a los de las máquinas más populares. Para cargar estos juegos, primero conecte su receptor de TV. Lo siguiente es APAGAR su Commodore 64. **DEBE APAGAR SIEMPRE SU COMMODORE 64 ANTES DE INSERTAR O EXTRAER UN CARTUCHO, SI NO LO HACE ASI PUEDE DAÑAR EL CARTUCHO Y/O SU COMMODORE.** Tercero, inserte, el cartucho. Ahora conecte su Commodore 64. Finalmente pulse la secuencia de teclas de inicio como se especifica en la hoja de instrucciones que encontrará con cada juego o siga las instrucciones que aparezcan en pantalla.



2. **CASSETTES:** Use su grabador DATASSETTE y la cinta ordinaria de audio que esté pregrabada. Asegúrese de que el cassette está completamente rebobinado al principio de la primera cara. Entonces escriba LOAD. El ordenador responderá con PRESS PLAY ON TAPE, pulse la tecla "PLAY" de su lector DATASSETTE. En este punto la pantalla del ordenador se borrará hasta que el programa se haya encontrado. El ordenador dirá FOUND (NOMBRE PROGRAMA) en la pantalla. Ahora pulse la tecla C= . Esto cargará el programa en el ordenador. Si quiere detener la operación de carga simplemente pulse la tecla RUN/STOP.

3. **DISCOS:** Para usar su unidad de discos, inserte cuidadosamente el disco preprogramado de manera que la etiqueta del disco mire hacia arriba y esté en la parte cercana a Vd. Compruebe la situación de una pequeña muesca en el disco (puede estar tapada con un trozo de cinta). Si inserta el disco debidamente la muesca estará en el lado izquierdo. Una vez el disco está dentro cierre la puerta de protección bajando la palanca. Ahora escriba LOAD "NOMBRE PROGRAMA", 8 y pulse la tecla RETURN . El disco girará y en su pantalla aparecerá:

```
SEARCHING FOR NOMBRE PROGRAMA
LOADING

READY
█
```

Cuando READY haya aparecido y esté el cursor, escriba RUN y pulse RETURN , su software pregrabado empezará a ser ejecutado.

CARGA DE PROGRAMAS DESDE CASSETTE

Cargar un programa desde cassette o disco es sencillo. Para cassette, rebobine la cinta hasta el principio y escriba:

```
LOAD "NOMBRE PROGRAMA"
```

Si no recuerda el nombre del programa, escriba LOAD y el primer programa de la cinta se cargará en memoria.

Después de pulsar RETURN el ordenador responde con:



```
PRESS PLAY ON TAPE
```

Después de que presione la tecla "PLAY", la pantalla se borrará pasando al color del borde de la pantalla mientras el ordenador busca el programa.

Cuando el programa se ha encontrado, la pantalla presentará:

```
FOUND NOMBRE PROGRAMA
```

Para que se cargue el programa, pulse la tecla C= . Para abandonar el procedimiento de carga, pulse RUN/STOP .Si pulsa la tecla Commodore, la pantalla volverá al color del borde mientras se carga el programa. Después de que el procedimiento de carga se ha completado, la pantalla vuelve a la normalidad y pronto reaparecerá READY.

CARGA DE PROGRAMAS DESDE DISCO

Cargar un programa desde disco sigue el mismo formato. Escriba:

```
LOAD "NOMBRE PROGRAMA",8
```

Después de pulsar RETURN el disco empezará a girar y la pantalla presentará:

```
SEARCHING FOR NOMBRE PROGRAMA
LOADING

READY
█
```


NOTA:

Cuando carga un nuevo programa en la memoria del ordenador, algun programa existente en la memoria previamente se borrará. Asegúrese de guardar el programa en el que ha trabajado antes de cargar uno nuevo. Una vez un programa se ha cargado, puede hacer RUN, LIST, o realizar cambios y grabar la nueva versión.

GRABACION DE PROGRAMAS EN CINTA

Después de entrar un programa, si desea guardarlo en una cinta, escriba:

```
SAVE "NOMBRE PROGRAMA"
```

"NOMBRE PROGRAMA" puede ser una combinación de un máximo de 16 caracteres. Después de pulsar RETURN el ordenador respondrá con:

```
PRESS PLAY AND RECORD ON TAPE
```

Presione las teclas "REC" y "PLAY" del DATASSETTE. La pantalla se borra, canviando al color del borde.

Después de que el programa se ha grabado en cinta reaparecerá READY, indicando que puede empezar a trabajar en otro programa, o apagar el ordenador sin perder su programa.

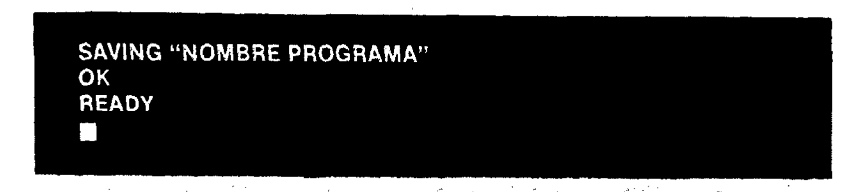
GRABACION DE PROGRAMAS EN DISCO

Guardar un programa en disco es igual de simple. Escriba:

```
SAVE "NOMBRE PROGRAMA",8
```

El 8 es el código de periférico de la unidad de disco, así el ordenador sabe que se quiere guardar el programa en disco.

Después de pulsar RETURN el disco empezará a girar y el ordenador responderá con:



PRINT Y CALCULOS

Ahora que ha asimilado gracias a unos ejemplos como grabar y leer programas, empezaremos a escribir algunos para guardarlos.

Pruebe escribiendo lo siguiente tal como se ve:



Si ha cometido un error de escritura, use la tecla INST/DEL para borrar el caracter inmediatamente a la izquierda del cursor. Puede borrar muchos caracteres si es necesario.

Veamos que ha pasado en el ejemplo anterior. Primero, se indica al ordenador que haga un PRINT (imprima, literalmente) con lo que esté dentro de las comillas. Pulsando RETURN se indica al ordenador que ejecute la instrucción y COMMODORE 64 se exhibe en la pantalla.

Cuando se utiliza la sentencia PRINT de esta forma, cualquier cosa encerrada entre comillas se exhibe exactamente como ha sido escrita.

Si el ordenador responde con:

?SYNTAX ERROR

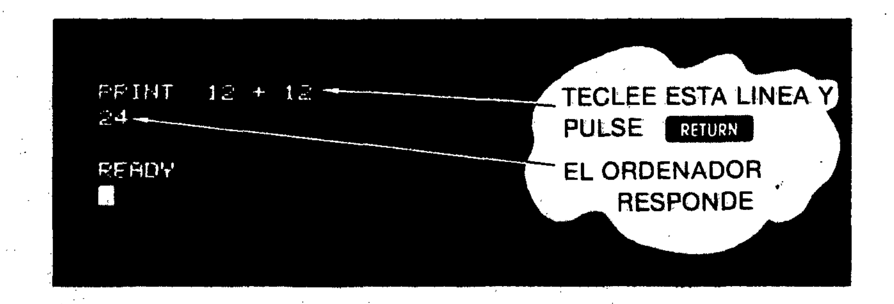
le anuncia que ha cometido un error de escritura, o se ha olvidado de las comillas. El ordenador admite solo instrucciones dadas de una forma específica.

Pero no se preocupe; recuerde entrar las cosas como las presentamos en los ejemplos y no tendrá problemas con el manejo de su Commodore 64.

Recuerde, no puede estropear el ordenador con errores de escritura y la mejor manera de aprender BASIC es probando diferentes instrucciones y observando los resultados.

PRINT es uno de los comandos más utilizados y potentes del lenguaje BASIC. Con él, puede presentar lo que desee, incluyendo gráficos, y resultados de cálculos.

Por ejemplo, pruebe lo siguiente. Limpie la pantalla pulsando las teclas SHIFT y CLR/HOME y escriba (asegúrese que utiliza la tecla "1", no la letra "I"):



Ha descubierto que el Commodore 64 es una calculadora en su forma básica. El resultado "24" ha sido calculado y presentado automáticamente. De hecho, puede también ejecutar restas, multiplicaciones, exponenciaciones, y funciones matemáticas complejas tales como calcular raíces cuadradas, etc. Y no se limita a un sólo cálculo en una línea, pero más adelante lo veremos con más detalle.

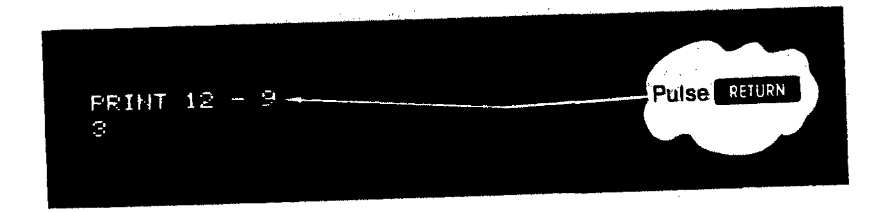
Nótese también que en la forma anterior, PRINT se comporta de manera diferente del primer ejemplo. En este caso, un valor o resultado de un cálculo se exhibe, en vez del mensaje exacto que se entró porque las comillas se omitieron.

SUMA

El signo más (+) señala la suma: le indica al ordenador que presente el resultado de 12 sumado a 12. Otras operaciones aritméticas toman una forma similar a la suma. Recuerde siempre pulsar RETURN después de escribir PRINT y el cálculo.

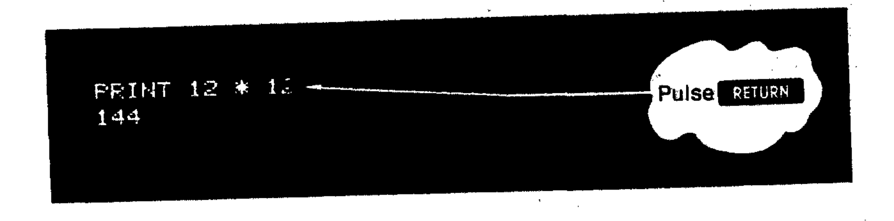
RESTA

Para restar, usar el signo convencional menos (—). Escriba:



MULTIPLICACION

Si quiere multiplicar 12 por 12, use el asterisco (*) para representar el signo de la multiplicación. Escribir:



DIVISION

La división usa la familiar "/". Por ejemplo, para dividir 144 por 12, escriba:

```
PRINT 144/12
12
```

Pulse RETURN

EXPONENCIACION

De manera similar, puede fácilmente elevar un número a una potencia (es lo mismo que multiplicar un número por sí mismo un número de veces especificado). La "↑" (Flecha arriba) significa exponenciación.
Escriba:

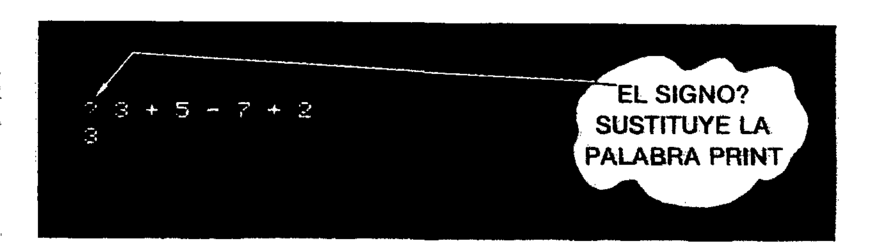
```
PRINT 12 ↑ 5
248832
```

Es lo mismo que escribir:

```
PRINT 12 * 12 * 12 * 12 * 12
248832
```

> **NOTA:**
>
> En BASIC se dispone de ciertas maneras de abreviar las operaciones. Una de tales maneras es abreviando los comandos BASIC (o palabras-clave). Se puede usar el signo "?" en lugar de PRINT, por ejemplo. El Apéndice D presenta las abreviaciones para cada uno y cómo se verán en la pantalla cuando se escriba la forma abreviada.

El último ejemplo presenta otro punto importante: muchos cálculos pueden ejecutarse en la misma línea, y pueden ser de varios tipos.
Puede calcular esta fórmula:



Hasta este momento ha utilizado números pequeños y ejemplos simples. De cualquier modo, el Commodore 64 es capaz de cálculos más complejos.

Puede, por ejemplo, sumar un número de muchas cifras. Pruebe esto, pero no use ninguna coma, o incurrirá en error.

```
? 123.45 + 345.78 + 7895.687
8364.917
```

Esto parece claro, pero ahora pruebe esto:

```
? 12123123.45 + 345.78 + 7895.687
12131364.9
```

Si se toma la molestia de sumar esto a mano, le dará un resultado diferente.

¿Qué ha pasado aquí? Igual que el ordenador tiene una potencia, hay un límite de los números que puede manejar. El Commodore 64 puede trabajar con números que contengan 10 dígitos. De cualquier modo cuando un número es presentado, sólo salen nueve dígitos.

Así en este ejemplo, el resultado es "redondeado" para adaptarse a su propio rango. El Commodore 64 redondea hacia arriba cuando el último dígito es cinco o más; redondea hacia abajo cuando el último dígito es cuatro o menos.

Números entre 0.01 y 999,999,999 se presentan utilizando la notación estandar. Números fuera de estos límites se exhiben utilizando la notación científica.

La notación científica es el procedimiento de expresar un número muy grande o muy pequeño como una potencia de 10.

Si escribe:

```
? 123000000000000000
1.23E+17
```




Esto es lo mismo que 1.23 * 10 ↑17 y se usa para manejar cantidades grandes.

Hay un límite para los números que el ordenador puede manejar, también en notación científica. Estos límites son:

NUMERO MAYOR: ± 1.70141183E+38
NUMERO MENOR: ± 2.93873588E—39

PRECEDENCIA

Si prueba a ejecutar algunos cálculos mixtos diferentes de los ejemplos que hemos presentado anteriormente, puede que no le den los resultados que esperaba. La razón es que el ordenador ejecuta los cálculos en un cierto orden.

En este cálculo:

20 + 8 / 2

no se puede decir que sea la respuesta 24 o 14 hasta conocer en qué orden se ejecutan las operaciones. Si suma 20 al resultado de dividir 8 entre 2, el resultado es 24. Pero, si suma 20 más 8 y divide el resultado por 2, la respuesta es 14. Pruebe el ejemplo y vea cual es el resultado.

La razón de que sea 24 es porque el Commodore 64 ejecuta las operaciones de izquierda a derecha de acuerdo con el siguiente orden:

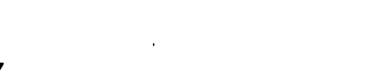
Primero:	—	el signo menos indica número negativo
Segundo:	↑	exponenciación, izquierda a derecha
Tercero:	*/	multiplicación y división, izquierda a derecha
Cuarto:	+ —	suma y resta, izquierda a derecha

Experimente con el orden de precedencia, y verá que en el ejemplo de arriba la división se ejecuta primero y entonces la operación da el resultado de 24. Haga algunas pruebas por sí mismo y vea si puede seguir adelante y predecir los resultados de acuerdo con las reglas dadas arriba.

Existe una manera sencilla de alterar el proceso de precedencia usando paréntesis para compensar las operaciones que quiera que se ejecuten primero.

Por ejemplo, si quiere dividir 35 por 5 más 2 escriba:

```
? 35 / 5 + 2
9
```

tendrá 35 dividido por 5 con 2 sumando al resultado, lo cual no es lo que se había propuesto. Para tener lo que realmente queríamos, pruebe esto:

```
? 35 / (5 + 2)
5
```

Lo que ha pasado ahora es que el ordenador ha evaluado primero lo contenido en los paréntesis. Si hay paréntesis entre paréntesis, se evalua primero lo contenido en los paréntesis interiores.

Cuando hay varios paréntesis en una línea, tal como:

```
? (12 + 9) * (6 + 1)
147
```

el ordenador evalua de izquierda a derecha. 21 se multiplica por 7 para obtener como resultado 147.

COMBINANDO COSAS

Igual que hemos empleado un montón de tiempo en áreas que pueden no parecerle muy importantes. los detalles presentados aquí le serán de gran utilidad una vez empiece a programar, y se revelarán de un valor inestimable.

Para darle una cierta idea de que sitio ocupa cada cosa, considere lo siguiente: ¿Cómo combinaría los dos tipos de sentencias PRINT que ha examinado de modo que se exhiba algo más significativo en la pantalla?

Vd. sabe que encerrando algo entre comillas imprime la información en la pantalla exactamente como se ha tecleado y utilizando operadores matemáticos. se ejecutan los cálculos. De este modo. ¿Por qué no combina los dos tipos de sentencias PRINT de manera similar a ésta?

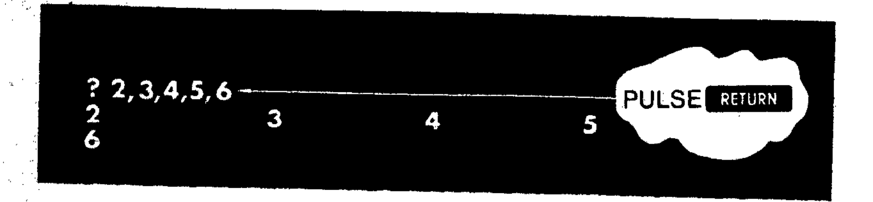
PUNTO Y COMA SIGNIFICA A CONTINUACION. SIN ESPACIO

```
? "5 * 9 = "; 5 * 9
5 * 9 = 45
```


Aunque esto puede parecer un poco redundante, hemos hecho uso de ambos tipos de sentencias PRINT unidas. La primera parte imprime "5 * 9 =" exactamente como se ha escrito. La segunda parte hace el cálculo y presenta el resultado, con el punto y coma se separa la parte del mensaje de la sentencia para calcular.

Puede separar las partes de una sentencia de impresión mixta con signos de puntuación para varios formatos. Pruebe una coma en lugar del punto y coma y vea qué pasa.

Para los curiosos, el punto y coma hace que la parte próxima de la sentencia sea exhibida inmediatamente después de la parte previa, sin ningún espacio. La coma lo hace algo diferente. Es igualmente un separador aceptable, espacia más las cosas. Si escribe:



los números se imprimirán de una parte a otra de la pantalla y abajo en la línea siguiente.

La presentación del Commodore 64 está organizada en 4 áreas de 10 columnas cada una. La coma tabula cada resultado en la próxima área disponible. Como pedimos que se exhiba más información que la que cabe en una línea, (probemos de adaptar 5 áreas de 10 columnas en una línea) el último valor se presentará en la línea siguiente.

CAPÍTULO

EL SIGUIENTE PASO

Hasta ahora hemos ejecutado algunas operaciones sencillas entrando una línea sola de instrucciones en el ordenador. Una vez se ha pulsado RETURN, la operación especificada se ejecuta inmediatamente. Esto se llama "MODO INMEDIATO o CALCULADOR".

Pero para hacer algo significativo, debemos poder operar con el ordenador con más de una línea de programa. Un número de sentencias combinadas unidas se llama un PROGRAMA y permite usar toda la potencia del Commodore 64.

Para que vea lo fácil que es escribir su primer programa en el Commodore 64, pruebe esto:

Borre la pantalla presionando la tecla SHIFT, y la tecla CLR/HOME.
Escriba NEW y pulse RETURN. (Esto borrará cualquier número que pudiera haber en la memoria del ordenador durante los anteriores experimentos).
Ahora escriba lo siguiente exactamente como se ilustra (Recuerde pulsar RETURN después de cada línea).

```
10 ?"COMMODORE 64"
20 GOTO 10
█
```

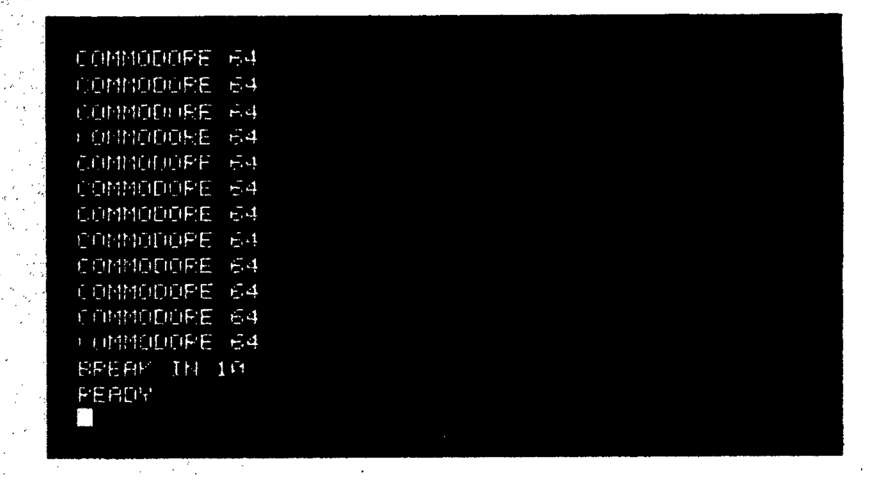
Ahora, escriba RUN y pulse RETURN, observe lo que pasa. Su pantalla empezará a llenarse con las palabras COMMODORE 64. Después de unos instantes pulse RUN/STOP para detener el programa.

Cierto número de conceptos importantes se han introducido en este corto programa: son las bases de toda programación.

Advertimos que aquí cada sentencia va precedida de un número. Este número de LINEA dice al ordenador en qué orden ha de ejecutar cada instrucción. Estos números son también un punto de referencia, en el caso de que el programa necesite retroceder a una línea determinada. Los números de línea pueden tener cualquier valor numérico (entero) entre 0-63999.

```
1Ø PRINT "COMMODORE 64"
↑    └── SENTENCIA O INSTRUCCIÓN
└──────── NUMERO DE LINEA
```



Una buena costumbre al programar es la de númerar las líneas en incrementos de 10 así en el caso de que se necesite insertar mas tarde otras líneas puede hacerse con los números que han quedado libres.

Además de PRINT, su programa también usa otra instrucción de BASIC: GOTO (ir a, en inglés). Esta hace que el ordenador vaya a una línea particular y la ejecute, entonces continua desde ese punto.

```
┌→ 1Ø PRINT "COMMODORE 64"
│
└─ 2Ø GOTO 1Ø
```

En el ejemplo, el programa imprime el mensaje en la línea 10, va a la línea siguiente (20), la cual le indica que debe volver a la línea 10. Así el ciclo se repite y no tendremos una manera de salir de este bucle, el programa seguirá eternamente hasta que lo paremos físicamente con la tecla RUN/STOP.

Una vez haya parado el programa, escriba: LIST. Su programa se presentará intacto, porque está en memoria. Advierta que el ordenador ha convertido el ? en PRINT para el listado. El programa ahora puede cambiarse, guardarse, o ponerse en marcha otra vez.

Otra diferencia importante entre escribir algo en modo inmediato y en modo programa es que una vez ejecutada y borrada la pantalla en una sentencia inmediata, ésta se ha perdido. Sin embargo, siempre puede recuperar el programa escribiendo LIST.

De esta manera, cuando use abreviaturas no olvide que el ordenador puede exceder la capacidad de una línea si usa demasiadas.



NOTAS EN LA EDICION DE PROGRAMAS

Si comete un error en una línea, tiene una serie de facilidades de edición para corregirlo.

1. Puede volver a escribir una línea, y el ordenador automáticamente sustituirá la antigua por la nueva.
2. Una línea que no se necesita puede borrarse simplemente escribiendo el número de línea y RETURN .
3. Puede también cambiar fácilmente una línea existente, usando las teclas de cursor y edición.

Supongamos que comete un error en una línea del ejemplo. Para corregirla sin tener de escribir de nuevo la línea entera, pruebe esto:

Escriba LIST, entonces use las teclas SHIFT y ⇑ CRSR ⇓ unidas para mover el cursor hacia arriba hasta posicionarlo en la línea que necesita cambiarse.

Ahora, utilice la tecla del cursor a la derecha para mover el cursor hacia el caracter que quiere cambiar, escriba el cambio encima del caracter antiguo. Ahora pulse RETURN y la línea correcta reemplazará a la antigua.

Si necesita más espacio en la línea, posicione el cursor donde se necesita el espacio y pulse SHIFT y INST/DEL al mismo tiempo y se abrirá un espacio. Ahora escriba la información adicional y pulse RETURN . Igualmente, puede borrar los caracteres que quiera situando el cursor a la derecha del caracter a borrar y pulse la tecla INST/DEL .

Para verificar estos cambios que se han entrado, escriba LIST otra vez, y se presentará el programa correcto. Las líneas que no se han entrado en orden numérico, el ordenador las situará automáticamente en su sitio dentro de su secuencia.

Pruebe a editar el programa simple de la página 33 cambiando la línea 10 y añadiendo una coma al final de la línea.

Entonces haga RUN otra vez.

10 PRINT"COMMODORE ",   ←—— NO OLVIDE MOVER EL CURSOR HASTA DEBAJO DE LA LINEA 20 ANTES DE EJECUTAR EL PROGRAMA.

Las variables son una de las características más utilizadas de cualquier lenguaje de programación, porque las variables pueden almacenar mucha información en el ordenador. Comprender cómo trabajan las variables será una gran ayuda para Vd. y nos permitirán realizar cosas que de otra manera no serían posible.




```
COMMODORE  COMMODORE  COMMODORE  COMMODORE
COMMODORE  COMMODORE  COMMODORE  COMMODORE
COMMODORE  COMMODORE  COMMODORE  COMMODORE
COMMODORE  COMMODORE  COMMODORE  COMMODORE
COMMODORE  COMMODORE  COMMODORE  COMMODORE
COMMODORE  COMMODORE  COMMODORE  COMMODORE
COMMODORE  COMMODORE  COMMODORE  COMMODORE
COMMODORE  COMMODORE  COMMODORE  COMMODORE
COMMODORE  COMMODORE  COMMODORE  COMMODORE
COMMODORE  COMMODORE  COMMODORE  COMMODORE
COMMODORE  COMMODORE  COMMODORE  COMMODORE
COMMODORE  COMMODORE  COMMODORE  COMMODORE
COMMODORE  COMMODORE  COMMODORE  COMMODORE
BREAK IN 10
READY
█
```

Imagínese un número de casilla dentro del ordenador que puede cada una contener un número o una cadena de caracteres de texto. Cada una de estas cajas está etiquetada con un nombre que escogemos. Este nombre se llama variable y representa la información de la caja respectiva.

Por ejemplo, si entramos:

```
10 X% = 15
20 X = 23.5
30 X$ = "LA SUMA DE X% + X = "
```

El ordenador representa las variables de ésta manera:

X% 15

X 23.5

X$ LA SUMA DE X% + X =

Un nombre de variable representa la caja donde está almacenado el valor actual de la variable. Como puede ver se puede asignar a una variable un número entero, un número en coma flotante, o una cadena de caracteres. El símbolo % que sigue a un nombre de variable indica que ésta le representará un número entero. Los siguientes son nombres válidos de variables enteras:



A%
X%
A1%
NM%

El "$" siguiente a un nombre de variable indica que ésta representa una cadena de caracteres. Lo siguiente son ejemplos de variable de cadena:

A$
X$
MI$

Las variables de coma flotante siguen el mismo formato, sin ninguna indicación:

A1
X
Y
MI

Para asignar un nombre a una variable hay algunas cosas a tener en cuenta. Primero, un nombre de variable puede tener uno o dos caracteres. El primer caracter debe ser un caracter alfabético de la A a la Z; el segundo caracter puede ser alfabético o numérico (en el rango 0 a 9). El tercer caracter puede incluirse para indicar el tipo de variable (entera o cadena de texto), % o $.

Pueden usarse nombres de variables que tengan mas de dos caracteres alfabéticos, pero sólo los dos primeros son reconocidos por el ordenador. Así PA y PARTNO son los mismos y se referirán a la misma caja de variable.

La última regla para nombres de variable es simple: no pueden contener ninguna palabra-clave BASIC (palabras reservadas) tales como GOTO, RUN, etc. Consulte el Apéndice D que contiene una lista completa de palabras reservadas BASIC.

Para ver cómo las variables pueden utilizarse para trabajar, escriba el programa que mencionamos antes y haga RUN. Recuerde pulsar RETURN después de cada línea en el programa.

```
NEW
1Ø X% = 15
2Ø x = 23.5
3Ø X$ = "LA SUMA DE X% + X = "
4Ø PRINT "X% = "; X%, "X = "; X
5Ø PRINT X$; X% + X
```




Si ha hecho cada cosa como se ve en la ilustración éste será el resultado que aparecerá en la pantalla:

```
RUN
X% = 15    X = 23.5
LA SUMA DE X% + X = 38.5
READY
■
```

Hemos reunido todos los trucos aprendidos así ahora para formatear la presentación como la ve e imprimir la suma de las dos variables.

En las líneas 10 y 20 asignamos un valor entero a X% y asignamos un valor en coma flotante a X. Esto coloca los números asociados con las variables en sus cajas. En la línea 30, asignamos una cadena de texto a X$. La línea 40 combina los dos tipos de sentencias PRINT para imprimir el mensaje y los valores actuales de X% y X. La línea 50 imprime la cadena de texto asignada a X$ y la suma de X% y X.

Nótese que al igual que X se usa como parte de cada nombre de variable, los identificadores % y $ hacen único, X%, X y X$, así representan tres variables distintas.

Pero las variables son mucho más potentes. Si cambia su valor, el nuevo valor sustituye al valor original en la misma caja. Esto permite escribir una sentencia similar:

X = X + 1

Esto no sería aceptado en álgebra normal, pero es uno de los conceptos más utilizados en programación. Esto significa: coge el valor actual de X, súmale uno y sitúa la nueva suma en la caja que representa X.

IF...THEN

Armados de la habilidad para actualizar fácilmente el valor de las variables, podemos ahora probar un programa tal como:

```
NEW
10 CT = 0
20 ?"COMMODORE 64"
30 CT = CT + 1
40 IF CT < 5 THEN 20
50 END
RUN
COMMODORE 64
COMMODORE 64
COMMODORE 64
COMMODORE 64
COMMODORE 64
```

Hemos introducido dos nuevos comandos BASIC, para proporcionar control sobre el pequeño programa que llenaba la pantalla con el mensaje "COMMODORE 64" presentado al inicio de este capítulo.

IF...THEN añade alguna lógica al programa. Dice IF (SI condicional, en inglés) una condición se cumple THEN (ENTONCES) realiza la operación. SI (IF) la condición no se cumple, ENTONCES (THEN) se ejecutará la próxima línea de programa.

Un cierto número de condiciones pueden explorarse usando una sentencia IF...THEN:

SIMBOLO	*SIGNIFICADO*
<	Menor que
>	Mayor que
=	Igual A
<>	No Igual
>=	Mayor o Igual Que
<=	Menor o Igual Que

El uso de alguna de estas condiciones es fácil, y sorprendentemente potente.

```
   10 CT = 0
┌→ 20 ?"COMMODORE 64"
│  30 CT = CT + 1
└─ 40 IF CT < 5 THEN 20
   ↓
   50 END
```




En este programa, hemos establecido un "bucle" que tiene una condición que cumplir: SI un valor es menor que 5 ENTONCES volver a 20.

La línea 10 coloca el valor de CT (CuenTa) a 0. La línea 20 imprime su mensaje. La línea 30 suma uno a la variable CT. Esta línea cuenta cuantas veces se repite el bucle. Cada vez que el bucle se ejecuta, CT aumenta uno.

La línea 40 es la línea de control. Si CT es menor que 5, significa que ha ejecutado el bucle menos de 5 veces, el programa vuelve a la línea 20 y la imprime de nuevo. Cuando CT es igual a 5 -indicando 5 impresiones COMMODORE 64- el programa va a la línea 50, la cual señala el fin (END) del programa.

Pruebe el programa y vea qué significa. Cambiando el límite 5 en la línea 40 puede tener cualquier número de líneas impresas.

IF...THEN tiene multitud de otros usos, los cuales se verán en futuros ejemplos.

BUCLES FOR...NEXT

Existe una manera más simple y eficaz de realizar la tarea del ejemplo previo y es usando un bucle FOR...NEXT. Considere lo siguiente:

```
NEW

10 FOR CT = 1 TO 5
20 PRINT "COMMODORE 64"
30 NEXT CT

RUN
COMMODORE 64
COMMODORE 64
COMMODORE 64
COMMODORE 64
COMMODORE 64
```

Como puede ver, el programa es más corto y más directo.

CT toma al principio el valor 1 en la línea 10. En la línea 20 exhibe el mensaje.



En la línea 30 CT se incrementa en 1. La sentencia NEXT (SIGUIENTE, en inglés) en la línea 30 automáticamente envía al programa a la línea 10 donde está la parte FOR (PARA) de la sentencia FOR...NEXT. Este proceso continuará hasta que CT alcance el límite señalado por TO en la línea 10.

La variable usada en el bucle FOR...NEXT puede incrementarse en unidades menores que 1, si es necesario.

Pruebe esto:

```
NEW

10 FOR NB = 1 TO 10 STEP .5
20 PRINT NB,
30 NEXT NB

RUN
1           1.5         2           2.5
3           3.5         4           4.5
5           5.5         6           6.5
7           7.5         8           8.5
9           9.5         10
```

Si entra y ejecuta este programa, verá que los números del 1 al 10, se exhiben a lo largo de la pantalla, en incrementos de 0,5.

Lo que estamos haciendo es exhibir los valores que va tomando NB a medida que avanza el bucle.

Puede igualmente especificar si la variable se incrementa o decrementa. Cambie la línea 10 por lo siguiente:

1ø FOR NB = 1ø TO STEP —.5

Y vea que ocurre lo contrario, como NB va de 10 a 1 en orden descendente.



INTRODUCCION

Los próximos capitulos se han escrito para los usuarios que están relativamente familiarizados con el lenguaje de programación BASIC y los conceptos necesarios para escribir programas en este lenguaje.

Aquellos que empiezan a aprender cómo se programa, pueden encontrar esta información quizas excesivamente técnica para comprenderla completamente. Pero tómeselo en serio...porque para estos dos divertidos capítulos, GRAFICOS SPRITE y CREANDO SONIDO, hemos puesto algunos ejemplos sencillos que están escritos para el usuario que empieza. Los ejemplos le darán una idea de cómo utilizar las sofisticadas capacidades sonoras y gráficas disponibles en su Commodore 64.

Si decide que quiere aprender más sobre preparación de programas en BASIC, hemos incluido una bibliografía (ver apéndice N) al final de este manual.

Si está dispuesto a familiarizarse con la programación BASIC, estos capítulos le ayudarán a empezar con técnicas avanzadas de programación BASIC. Información más detallada se puede encontrar en la GUIA DE REFERENCIA DEL PROGRAMADOR DEL COMMODORE 64, disponible a través de su distribuidor local de Commodore.

ANIMACION

Ejercitaremos ahora algunas de las capacidades gráficas del Commodore 64 uniendo lo que hemos visto hasta ahora, junto con nuevos conceptos. Si es audaz, escriba el siguiente programa y vea lo qué pasa. Advertirá que dentro de las sentencias de impresión pueden también incluirse controles de cursor y comandos de pantalla. Cuando vea la imagen de una tecla en un listado de programa, pulse la tecla o combinación de teclas que correspondan. La pantalla mostrará la representación gráfica que está asignada a la función correspondiente.

```
NEW

10 REM PELOTA BOTANDO
20 PRINT"♥":REM"♥ ES SHIFT CLR/HOME
25 FORX=1TO10:PRINT"Q":NEXT:REM"Q ES CRSR ABAJO
30 FORBL=1TO40
40 PRINT" ●▌";:REM "● ES SHIFT Q;▌ ES CRSR IZQUIERDO;
   EL ESPACIO ES INTENCIONADO
50 FORTM=1TO5
60 NEXTTM
70 NEXTBL
75 REM MUEVE LA PELOTA DE DERECHA A IZQUIERDA
80 FORBL=40TO1STEP-1
90 PRINT" ▌▌●▌";:REM "▌ ES CRSR IZQUIERDO;● ES SHIFT Q;
   EL ESPACIO ES INTENCIONADO
100 FORTM=1TO5
110 NEXTTM
120 NEXTBL
130 GOTO20
```

: INDICACION DE NUEVOS COMANDOS

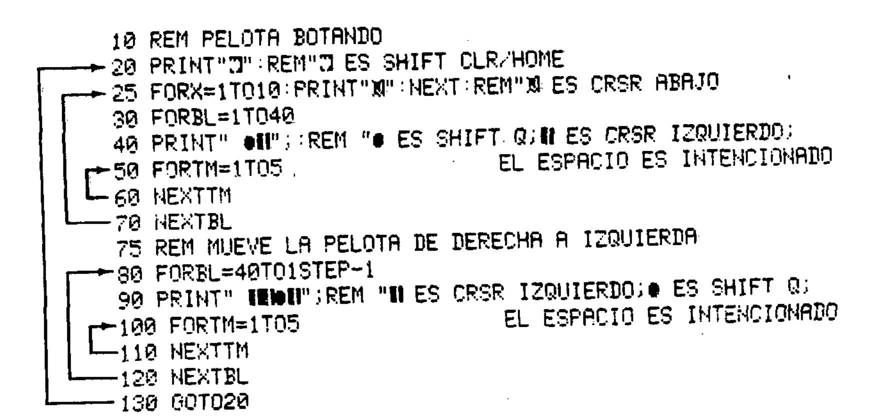
ESTOS ESPACIOS SON INTENCIONADOS

El programa mostrará una pelota rebotando y moviéndose de izquierda a derecha, y vuelve otra vez, por toda la pantalla.

Si observamos el programa de la página 44 podemos ver como se realiza esta acción.

La línea 10 es un comentario (REMark) que cuenta qué hace el programa;





```
10 REM PELOTA BOTANDO
20 PRINT"■":REM"■ ES SHIFT CLR/HOME
25 FORX=1TO10:PRINT"■":NEXT:REM"■ ES CRSR ABAJO
30 FORBL=1TO40
40 PRINT" ●■";:REM "● ES SHIFT Q;■ ES CRSR IZQUIERDO;
50 FORTM=1TO5                EL ESPACIO ES INTENCIONADO
60 NEXTTM
70 NEXTBL
75 REM MUEVE LA PELOTA DE DERECHA A IZQUIERDA
80 FORBL=40TO1STEP-1
90 PRINT" ■■●■";REM "■ ES CRSR IZQUIERDO;● ES SHIFT Q;
100 FORTM=1TO5               EL ESPACIO ES INTENCIONADO
110 NEXTTM
120 NEXTBL
130 GOTO20
```

no tiene en sí efecto sobre el programa. La línea 20 limpia la pantalla de información.

La línea 25 imprime 10 comandos cursor-abajo. Esto posiciona la pelota en medio de la pantalla. Si la línea 25 se eliminara la pelota se movería a lo largo de la línea superior de la pantalla.

La línea 30 inicia un bucle para mover la pelota las 40 columnas de izquierda a derecha.

La línea 40 hace muchas cosas. Lo primero se imprime un espacio para borrar la posición previa de la pelota, entonces imprime la pelota, y finalmente ejecuta un cursor-izquierda para estar listo para borrar la posición actual de la pelota otra vez.

El bucle puesto en las líneas 50 y 60 retarda la pelota un poco o demora el programa. Si no, la pelota se moverá demasiado rápido para verla.

La línea 70 completa el bucle que imprime pelotas en la pantalla, iniciado en la línea 30. Cada vez que el bucle se ejecuta, la pelota se mueve otro espacio a la derecha. Como advierte en la ilustración, se pone un bucle dentro de otro bucle.

Esto es perfectamente aceptable. Sólo le irá mal cuando un bucle se cruce con otro. Para ayudarle a escribir programas para probarlos lo hemos ilustrado aquí; asegúrese de que la lógica del bucle es correcta.

Para ver qué pasará si se cruzan dos bucles, invierta las sentencias de las líneas 60 y 70. Encontrará un error porque el ordenador se confunde y no puede saber dónde debe ir.

Las líneas 80 a 120 invierten los pasos de la primera parte del programa, y mueve la pelota de derecha a izquierda. La línea 90 es levemente diferente de la línea 40 porque la pelota se mueve en la dirección opuesta (hemos borrado la pelota a la derecha y hemos movido a la izquierda).



Cuando se ha ejecutado todo el programa vuelve a la línea 20 para empezar todo el proceso otra vez. Para parar el programa presione la tecla (RUN/STOP) y pulse (RESTORE).

Para una variación en el programa, edite la línea 40 para leer:

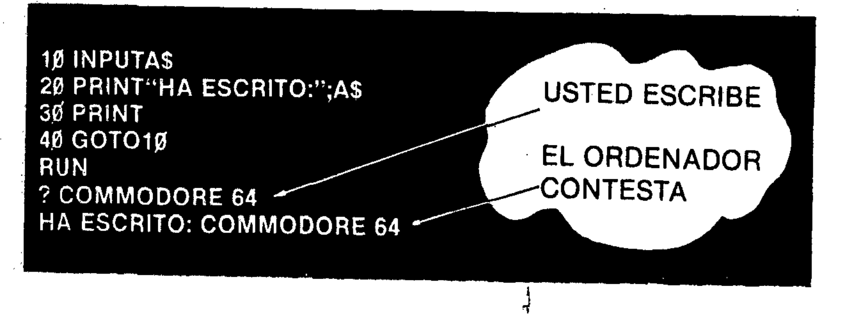
40 PRINT "●"; ← PARA HACER LA ●, PRESIONE LA TECLA SHIFT Y PULSE LA LETRA "Q".

Ejecute el programa y vea qué pasa ahora. Como dejamos fuera el control de cursor, cada pelota queda en la pantalla hasta que se borra por la pelota que se mueve de derecha a izquierda en la segunda parte del programa.

INPUT

Hasta ahora, cada cosa dentro de un programa se ha puesto antes de ejecutarlo. Una vez el programa está en marcha, no se podrá cambiar nada. INPUT (ENTRAR en inglés) nos permite el paso de nueva información al programa cuando se está ejecutando y actuar en función de esta información.

Para que tenga una idea de cómo funciona INPUT, escriba NEW RETURN y entre este corto programa.



Lo que pasa cuando se ejecuta este programa es sencillo. Aparecerá un signo de interrogación, indicando que el ordenador está esperando para que escriba algo. Entre algún caracter, o grupo de caracteres, por el teclado y pulse RETURN. El ordenador entonces responderá con "HA ESCRITO:" seguido de la información que se entró.

Esto puede parecer muy elemental, pero trate de imaginar lo qué puede hacer el ordenador con la información.

Puede entrar variables numéricas o de cadena, y se dispone además de un mensaje de aviso. El formato de INPUT es:

INPUT "MENSAJE DE AVISO";VARIABLE

EL AVISO DEBE SER MENOR DE 38 CARACTERES



O, solo:

INPUT VARIABLE

NOTA: Para salir de este programa pulse las teclas RUN/STOP y RESTORE .

El siguiente programa no es solo útil, muestra un montón de cosas que han sido presentadas hasta ahora, incluyendo las nuevas sentencias de entrada.

```
NEW

1 REM PROGRAMA DE CONVERSION DE TEMPERATURA
5 PRINT"♥":REM"♥ ES SHIFT CLR/HOME
10 PRINT"CONVIERTE EN FAHRENHEIT O CELSIUS (F/C)":INPUTA$
20 IFA$=""THEN10
30 IFA$="F"THEN100
40 IFA$<>"C"THEN10
50 INPUT"ENTRE GRADOS CELSIUS: ";C
60 F=(C*9)/5+32
70 PRINTC;" GRAD. CELSIUS = ";F;" GRAD. FAHRENHEIT
80 PRINT
90 GOTO10
100 INPUT"ENTRE GRADOS FAHRENHEIT: ";F
110 C=(F-32)*5/9
120 PRINTF;" GRAD. FAHRENHEIT = ";C;" GRAD.
    CELSIUS"
130 PRINT
140 GOTO10
```

AQUI NO HAY ESPACIO

NO SE OLVIDE DE PULSAR RETURN

Si entra y ejecuta este programa, verá a INPUT en acción.

La línea 10 usa la sentencia de entrada no solo para recoger información, sino que también exhibe un mensaje. Además podemos pedir un número o cadena alfanumérica (usando una variable númerica o de cadena).

Las líneas 20, 30 y 40 comprueban lo que se ha escrito. En la línea 20, si nada se entra (sólo se pulsa RETURN), entonces el programa vuelve a la línea 10 y pide otra vez la entrada. En la línea 30, si se escribe F, sabrá que el usuario quiere convertir la temperatura de grados Fahrenheit a Celsius, así el programa va a la parte donde se hace la conversión.

La línea 40 hace una prueba más. Sabemos que hay sólo dos opciones válidas que el usuario puede entrar. Para llegar a la línea 40, el usuario debe haber escrito algún caracter que no es F. Ahora, se hace una prueba para ver si este caracter es C; si no, el programa pide la entrada otra vez.



Esto puede parecer un detalle superfluo, pero es una buena práctica de programación.

Un usuario no familiarizado con el programa puede quedar muy frustrado si ocurre algo extraño porque un error ha hecho entrar información imprevista.

Una vez hemos determinado que tipo de conversión ejecutamos, el programa hace el cálculo e imprime la temperatura entrada y la convertida.

El cálculo es exacto matemáticamente, usando la fórmula establecida para conversión de temperatura. Después de que el cálculo se acaba y la respuesta se imprime, el bucle del programa vuelve a empezar.

Después de ejecutar la pantalla será similar a esta:

```
CONVIERTE EN FAHRENHEIT O CELSIUS (F/C): ?F
ENTRE GRADOS FAHRENHEIT: 32
32 GRADOS FAHRENHEIT = Ø GRADOS CELSIUS

CONVIERTE EN FAHRENHEIT O CELSIUS (F/C): ?
```

Después de ejecutar el programa, asegúrese de guardarlo en disco o cinta. Este programa, como muchos otros presentados a través de este manual, pueden formar parte de su biblioteca de programas.

GET

GET permite la entrada de un caracter cada vez desde el teclado sin pulsar RETURN . Esto realmente acelera la entrada de datos en muchas aplicaciones. Cualquier tecla pulsada es asignada a la variable que se especifica con GET.

La siguiente rutina ilustra como trabaja GET:

```
1 PRINT"♥":REM"♥ ES SHIFT CLR/HOME
10 GETA$:IFA$=""THEN10
20 PRINTA$;
30 GOTO10
```

NO HAY ESPACIO AQUI





Si hace RUN con este programa, la pantalla se limpiará y cada vez que pulse una tecla la línea 20 la imprimirá en la pantalla, y entonces hará un GET (tomará) otro caracter. Es importante notar que el caracter entrado no se mostrará a menos que especifique PRINT en la pantalla, como se ha hecho aquí.

La segunda sentencia de la línea 10 es también importante. GET trabaja contínuamente, igual si no se ha pulsado ninguna tecla (no como INPUT que espera una respuesta), así la segunda parte de esta línea continuamente prueba el teclado hasta que una tecla es pulsada.

Vea que pasa si la segunda parte de la línea 10 es eliminada.

Para parar el programa puede pulsar las teclas RUN/STOP y RESTORE.

La primera parte del programa de conversión de temperatura puede fácilmente escribirse usando GET. Cargue el programa de conversión de temperatura, y modifique las líneas 10, 20 y 40 como se muestra:

```
10 PRINT"CONVIERTE EN FAHRENHEIT O CELSIUS (F/C)"
20 GETA$:IFA$=""THEN20
30 IFA$<>"C"THEN20
```

Esta *modificación hará que el* programa opere suavemente, nada pasará a menos que el usuario pulse alguna de las teclas previstas.

Una vez está hecho el cambio, asegúrese de que guarda la nueva versión del programa.

NUMEROS ALEATORIOS Y OTRAS FUNCIONES

El Commodore 64 contiene un número de funciones que son usadas para ejecutar operaciones especiales. Las funciones pueden ser incluidas en un programa BASIC. Pero antes de que escriba un número de sentencias cada vez que necesite ejecutar un cálculo especializado, escriba el comando de la función deseada y el ordenador hará el resto.

Muchas veces cuando diseñamos un juego o un programa educativo, se necesita generar un número aleatorio, para simular la tirada de un dado, por ejemplo. Puede ciertamente escribir un programa que generará estos números, pero una vía fácil es llamar a la función de números RaNDom (aleatorios).

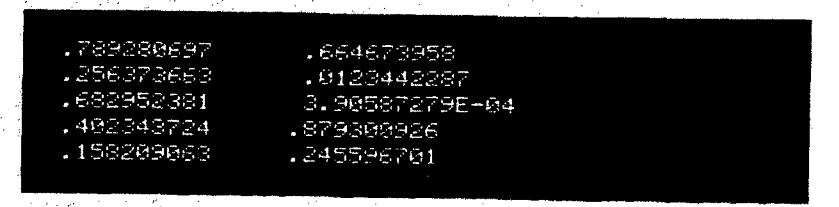



```
NEW

10 FORX=1TO10
20 PRINTRND(1),
30 NEXT
```

SI QUITA LA COMA SU LISTA DE NUMEROS APARECERA A 1 COLUMNA

Después de comenzar el programa, verá en la pantalla algo similar a esto:



¿Sus números no son iguales? ¡Bien, si se toma la molestia de comprobarlo serán completamente aleatorios!

Pruebe ejecutando el programa unas cuantas veces para verificar que los resultados son siempre diferentes. Igual si los números no siguen algún patrón, empezará por advertir que algunos salen a la vez que el programa empieza.

Primero, los resultados siempre están entre 0 y 1, pero nunca igual a 0 ó 1. Esto nunca será hecho ciertamente si queremos simular el lanzamiento al azar del dado, desde que estamos mirando números entre 1 y 6.

La otra característica importante que vemos es que estamos tratando con números reales (con cifras decimales). Esto puede ser también un problema *dado que a veces se necesitan números enteros.*

Hay maneras sencillas de producir números de la función RND en el rango deseado.

Sustituya la línea 20 por la siguiente y ejecute el programa otra vez:

```
20 PRINT 6*RND(1),

RUN

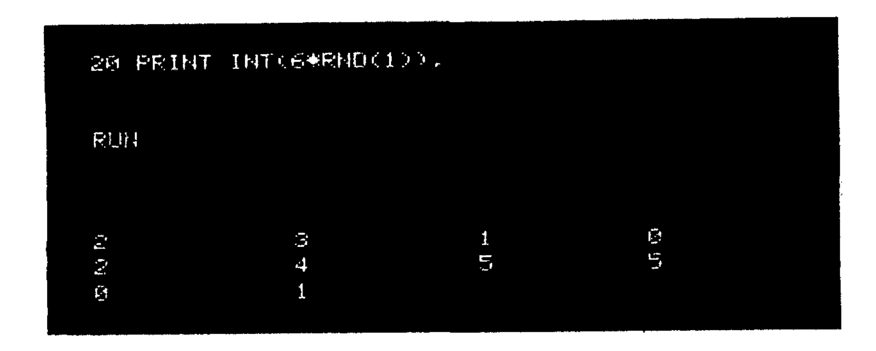

3.60563664      4.53660853
5.47238963      8.40850227
3.19265054      4.39547668
3.16331095      5.50620749
9.32527884      4.17090293
```




Esto resuelve el problema de no obtener resultados mayores que 1, pero aún tenemos que tratar el problema de la parte decimal del resultado.

La función INTeger (entero) convierte números reales en valores enteros.

Una vez más, sustituye la línea 20 por la siguiente y ejecute el programa para ver el efecto del cambio:



Esto nos permite generar número aleatorios entre 1 y 6. Si examina lo que hemos generado esta última vez, encontrará que el rango de los resultados va solo de 0 a 5.

Como último paso, sume un uno a la sentencia, como sigue:

2∅ PRINT INT(6*RND(1)) + 1

Ahora, hemos obtenido los resultados deseados.

En general, puede situar un número, variable, o alguna expresión BASIC entre los paréntesis de la función INT. Dependiendo del rango deseado, puede multiplicar el límite superior por la función RND. Por ejemplo, para generar números aleatorios entre 1 y 25, puede escribir:

2∅ PRINT INT(25*RND(1)) + 1

La fórmula general para generar un juego de números aleatorios en un rango fijo es:

NUMERO = INT (LIMITE SUPERIOR*RND(1)) + LIMITE INFERIOR

JUEGOS DE ADIVINACION

Después de dar algún rodeo para entender los números aleatorios, ¿por qué no utilizar esta información?. El siguiente juego no sólo ilustra un buen uso de los números aleatorios, sino que también introduce algunas teorías adicionales de programación.

Ejecutando este programa, un número aleatorio, NM, será generado.

INDICA NO ESPACIO DESPUES DE COMILLAS

```
NEW

1 REM JUEGO DE ADIVINAR NUMEROS
2 PRINT"♥":REM"♥ ES SHIFT CLR/HOME
5 INPUT"ENTRAR EL VALOR SUPERIOR PARA ADIVINAR";LI
10 NM=INT(LI*RND(1))+1
20 PRINT"YA SE EL NUMERO."
30 INPUT"CUAL ES MI NUMERO";GU
35 CN=CN+1
40 IFGU>NMTHENPRINT"MI NUMERO ES MENOR":PRINT:GOTO30
50 IFGU<NMTHENPRINT"MI NUMERO ES MAYOR":PRINT:GOTO30
60 IFGU=NMTHENPRINT"BRAVO! ESE ES MI NUMERO"
65 PRINT"EN SOLO ";CN;"INTENTOS.":PRINT
70 PRINT"QUIERE INTENTARLO DE NUEVO (S/N)";
80 GETAN$:IFAN$=""THEN80
90 IFAN$="S"THEN2
100 IFAN$<>"N"THEN80
110 END
```

Puede especificar cómo será de largo el número al empezar el programa. Entonces, adivine qué número es.

```
ENTRAR EL VALOR SUPERIOR PARA ADIVINAR? 25
YA SE EL NUMERO.

CUAL ES MI NUMEROZ? 15
MI NUMERO ES MAYOR.

CUAL ES MI NUMERO? 2∅
MI NUMERO ES MENOR.

CUAL ES MI NUMERO? 19
BRAVO ESTE ES MI NUMERO
EN SOLO 3 INTENTOS.

QUIERE INTENTARLO DE NUEVO (SIN)?
```

Las sentencias IF/THEN comparan su número con el número generado. Dependiendo de su número, el programa le dice si su número es más alto o bajo que el número aleatorio generado.

Para que la fórmula adquiera un determinado rango de número aleatorio, vea si puede añadir unas líneas del programa que permite al usuario también especificar el rango bajo de los números generados.

Cada vez que hace una jugada, CN se incrementa en 1 para guardar el número de jugadas. Usando el programa, vea si puede usar un buen razonamiento para hallar un número en el menor número de jugadas.

Cuando encuentre la respuesta correcta, el programa imprime el mensaje ¡"BRAVO! ESE ES MI NUMERO", con el número de intentos que ha hecho. Puede entonces empezar el proceso de nuevo. Recuerde, el programa genera un nuevo número aleatorio cada vez.

NOTAS DE PROGRAMACION

En las líneas 40 y 50, los dos puntos son usados para separar sentencias múltiples en una sola línea. Esto no salva de escribir, pero en programas largos conservará espacio en memoria.

También advierta en las sentencias IF/THEN en las mismas dos líneas, instruimos al ordenador que PRINT algo, inmediatamente después de bifurcar para algún otro punto del programa.

El último punto ilustra la razón del uso de los números de línea en incrementos de 10: Después de que el programa es escrito, decidimos añadir la parte contadora. Añadiendo esas nuevas líneas en el final del programa, numeradas para caer entre las líneas existentes, el programa es fácilmente modificado.

JUEGO DE DADOS

El siguiente programa simula la tirada de dos dados. Puede disfrutarlo como muestra, o usarlo como parte de un juego largo.

```
5 PRINT"QUIERE PROBAR SU SUERTE?"
10 PRINT"DADOS ROJOS = ";INT(6*RND(1))+1
20 PRINT"DADOS BLANCOS = ";INT(6*RND(1))+1
30 PRINT"PULSE LA BARRA ESPACIADORA PARA LA PROXIMA TIRADA":PRINT
40 GETA$:IFA$=""THEN40
50 IFA$=CHR$(32)THEN10
```

¿Quiere probar su suerte?

Para lo que ha aprendido sobre números aleatorios y BASIC, vea si puede seguir esto.



GRAFICOS ALEATORIOS

Como una nota final en la explicación de los números aleatorios, y como una introducción al diseño gráfico, entre y ejecute este pequeño programa:

```
10 PRINT "{CLR/HOME}"
20 PRINT CHR$(205.5 + RND(1));
40 GOTO 20
```

Como puede haber supuesto, la línea 20 es la clave aquí. Otra función, CHR$ (Cadena de Caracteres), toma un caracter, basado en un número standar de código de 0 a 255. Todo caracter del Commodore 64 puede imprimirse si está codificado de esta manera (vea Apéndice F).

Para buscar rápidamente el código de algún caracter, escriba:

PRINT ASC("X")

donde X es el caracter que se prueba (esto puede ser algún caracter imprimible, incluyendo gráficos). La respuesta es el código del caracter que ha escrito. Como probablemente se figura, ASC es otra función, que da el código standard ASCII del caracter que ha escrito.

Puede ahora imprimir este caracter escribiendo:

PRINT CHR$(X)

Si prueba escribiendo:

PRINT CHR$(205); CHR$(206)

verá los dos caracteres gráficos del lado derecho de las teclas M y N. Hay los dos caracteres en este programa usándose para el laberinto.

Usando la fórmula 205.5 + RND(1) el ordenador escogerá un número aleatorio entre 205.5 y 206.5. Hay un azar al 50% del número que es mayor o menor que 206. CHR$ ignora algún valor fraccionado, así la mitad del tiempo el caracter con código 205 es impreso y el resto del tiempo el código 206 es mostrado.

Si quiere experimentar con este programa, pruebe cambiando 205.5 por adiciones o sustracciones emparejándolo con décimas.



[illegible]

[illegible]

- [illegible]
- [illegible]
- [illegible]
- [illegible]
- [illegible]
- [illegible]
- [illegible]

COLOR Y GRAFICOS

Hasta ahora hemos explorado algunas de las sofisticadas posibilidades del Commodore 64, pero una de las más fascinantes es su descollante habilidad para producir color y gráficos.

Ha visto un rápido ejemplo de gráficos en la "pelota rebotante" y programas de "laberinto". Pero este sólo toca por encima la potencia del comando. Un número de nuevos conceptos se introducirán en esta sección para explicar la programación gráfica y de color y se mostrará cómo Vd. puede crear sus propios juegos y animación avanzada.

Porque se ha concentrado en las capacidades de cálculo de la máquina, todas las presentaciones han estado generadas hasta ahora en un solo color (texto azul claro en un fondo azul oscuro, con un borde azul claro).

En este capítulo verá cómo se añade color a los programas y el control de todos esos extraños símbolos gráficos del teclado.

[illegible] COLORES

Como ha descubierto si ha probado el test de alineación de color en el Capítulo 1, puede cambiar los colores de texto simplemente pulsando la tecla CTRL y una de las teclas de color. Esto trabaja bien en modo inmediato, pero ¿qué pasará si quiere incorporar cambios de color en sus programas?

Tiene un completo rango de 16 colores de texto para trabajar. Usando la tecla CTRL y una tecla numérica, los siguientes colores están disponibles:

1	2	3	4	5	6	7	8
Negro	Blanco	Rojo	Cián	Púrpura	Verde	Azul	Amarillo

Si pulsa la tecla C= con las teclas numéricas apropiadas, estos 8 colores adicionales pueden usarse:

1	2	3	4	5	6	7	8
Naranja	Marrón	Rosa	Gris 1	Gris 2	Verde Cl.	Azul Cl.	Gris 3

ESCRIBA NEW, y experimente con lo siguiente: Presione la tecla CTRL y al mismo tiempo pulse la tecla 1, a continuación pulse la letra C sin pulsar la tecla CTRL. Ahora, mientras presiona otra vez la tecla CTRL al mismo tiempo pulse la tecla [illegible]. Suelte la tecla CTRL y pulse la tecla [illegible]. Moviendo a través de los números, alternando con las letras, y escribiendo la palabra COLORES como sigue:

```
10 PRINT" C O L O R E S"
         ↑ ↑ ↑ ↑ ↑ ↑ ↑
   CTRL  1 2 3 4 5 6 7
```

RUN
COLORES

Igual que los controles de cursor muestran unos caracteres gráficos entre las comillas de las sentencias de impresión, los controles de color están también representados por caracteres gráficos.

En el ejemplo previo, cuando pulsa CTRL y escribe 3 un signo de libra se exhibe. CTRL 1 presentará una "←". Cada control de color presentará un único código gráfico cuando se usa de esta manera. Esta tabla muestra las representaciones gráficas para cada color imprimible.

TECLADO	COLOR	PRESENTACION	TECLADO	COLOR	PRESENTACION
CTRL 1	NEGRO		C= 1	NARANJA	
CTRL 2	BLANCO		C= 2	MARRON	
CTRL 3	ROJO		C= 3	ROJO CL.	
CTRL 4	CIAN		C= 4	GRIS 1	
CTRL 5	PURPURA		C= 5	GRIS 2	
CTRL 6	VERDE		C= 6	VERDE CL.	
CTRL 7	AZUL		C= 7	AZUL CL.	
CTRL 8	AMARILLO		C= 8	GRIS 3	

Igual que la sentencia PRINT puede verse un poco extraña en la pantalla, cuando ejecute el programa, solo el texto será presentado. Y automáticamente cambiarán los colores de acuerdo con los controles de color que situó en la sentencia de impresión.

Pruebe unos pocos ejemplos por sí mismo, mezclando algún número de colores dentro de una sentencia PRINT. Recuerde, además, que puede usar el segundo juego de colores de texto usando la tecla Commodore y las teclas numéricas.

AVISO:

Advertirá que después de ejecutar un programa con cambios de color o modo (inverso), hasta el aviso READY y algún texto adicional que escribió es el mismo que el último color o modo cambiado. Para volver a la presentación normal, recuerde pulsar:

RUN/STOP y RESTORE

CODIGOS CHR$ DE COLOR

Eche una mirada al Apéndice F, entonces vuelva a esta sección.

Puede haber advertido en una mirada sobre la lista de códigos CHR$ en el Apéndice F que cada color (como muchos controles de teclado, tal como los movimientos de cursor) tienen un código único. Estos códigos pueden imprimirse directamente obteniendo los mismos resultados que escribiendo CTRL y la tecla apropiada dentro de una sentencia PRINT.

Por ejemplo, pruebe esto:

```
NEW
10 PRINT CHR$(147) : REM" ▢ ES SHIFT CLR/HOME
20 PRINT CHR$(30);"CHR$(30) ME CAMBIA?"
RUN
CHR$(30) ME CAMBIA?
```

El texto sería ahora verde. En muchos casos, usar la función CHR$ será más fácil, especialmente si se quiere experimentar con el cambio de colores. En la página siguiente se dá una manera diferente de hacer el juego de colores. Puesto que hay un número de líneas que son similares (40-110) use las teclas de edición para evitar entrar todas las líneas otra vez. Vea las notas después del listado para refrescar su memoria en los procedimientos de edición.

```
NEW

1 REM BARRAS DE COLOR
5 PRINTCHR$(147):REM CHR$(147)=CLR/HOME
10 PRINTCHR$(18);"     ";:REM BARRA INVERTIDA
20 CL=INT(8*RND(1))+1
30 ONCLGOTO40,50,60,70,80,90,100,110
40 PRINTCHR$(5);:GOTO10
50 PRINTCHR$(28);:GOTO10
60 PRINTCHR$(30);:GOTO10
70 PRINTCHR$(31);:GOTO10
80 PRINTCHR$(144);:GOTO10
90 PRINTCHR$(156);:GOTO10
100 PRINTCHR$(158);:GOTO10
110 PRINTCHR$(159);:GOTO10
```




Escriba las líneas 5 a 40 normalmente. El aspecto de la pantalla sería similar a este:

```
1 REM BARRAS DE COLOR
5 PRINTCHR$(147):REM CHR$(147)=CLR/HOME
10 PRINTCHR$(18);"     ";:REM BARRA INVERTIDA
20 CL=INT(8*RND(1))+1
30 ONCLGOTO40,50,60,70,80,90,100,110
40 PRINTCHR$(5);:GOTO10
█
```

NOTAS DE EDICION

Use la tecla CRSR-ARRIBA para posicionar el cursor en la línea 40. Entonces escriba 5 sobre el 4 de 40. A continuación use la tecla CRSR-DERECHA para mover sobre el 5 en el paréntesis de CHR$. Pulse SHIFT INST/DEL para abrir un espacio y escriba "28". Ahora pulse RETURN con el cursor en cualquier parte de la línea.

Ahora la pantalla sería similar a esta:

```
NEW

1 REM BARRAS DE COLOR
5 PRINTCHR$(147):REM CHR$(147)=CLR/HOME
10 PRINTCHR$(18);"     ";:REM BARRA INVERTIDA
20 CL=INT(8*RND(1))+1
30 ONCLGOTO40,50,60,70,80,90,100,110
40 PRINTCHR$(28);:GOTO10
█
```

No se preocupe. La línea 40 está todavía ahí. LISTe el programa y vea. Usando el mismo procedimiento, continue para modificar la última línea con un nuevo número de línea y código CHR$. Como prueba final, liste el programa entero para asegurarse de que todas las líneas se han entrado debidamente antes de hacer RUN.

Aquí hay una corta explicación de lo que pasa.

Probablemente ha entendido la mayor parte del programa de las barras de color excepto por la extraña nueva sentencia de la línea 30. Pero rápidamente

veremos que hace el programa entero realmente. La línea 5 es el código de CLR/HOME.

La línea 10 pone en tipo inverso e imprime 5 espacios, que es una barra, toda está invertida. La primera vez a través del programa la barra será azul claro, el color normal de texto.

La línea 20 usa su función principal, la función aleatoria que selecciona un color al azar entre 1 y 8.

La línea 30 contiene una variante de la sentencia IF...THEN que es llamada ON...GOTO. ON...GOTO permite que el programa escoja una lista de números de línea a donde ir. Si la variable (en este caso CL) tiene un valor de 1, el primer número de línea es el que se escoge (aquí 40). Si el valor es 2, el segundo número de la lista es usado, etc.

Las líneas 40-110 convierten la clave de los colores aleatorios al código apropiado CHR$ para este color y retorna el programa a la línea 10 para hacer un PRINT de una sección de la barra en este color. Entonces el proceso entero empieza de nuevo.

Vea si puede figurarse cómo producir 16 colores aleatorios, expandir ON...GOTO para gobernarlos, y añadiendo los códigos CHR$ que quedan para presentar los 8 colores que quedan.

PEEKS Y POKES

No, no hablaremos sobre cómo hurgar en el ordenador, pero será posible "echar una mirada" dentro de la máquina y "sacar" cosas de ahí.

Como las variables pueden ser representadas por "casillas" en la máquina donde sitúa la información, puede también pensar en algunas "casillas" especiales en el ordenador que representan a posiciones de memoria específicas.

El Commodore 64 mira en esas posiciones de memoria para ver qué colores de fondo y borde de la pantalla cogerá, qué caracteres se presentarán en la pantalla —y dónde— y una multitud de otras tareas.

Situando, mediante POKE un valor diferente en la posición de memoria adecuada, podemos cambiar colores, definir y mover objetos, e igualmente crear música.

Esas posiciones de memoria pueden representarse de una manera similar a esta:

53280	53281	53282	53283
X	Y		
COLOR BORDE	COLOR FONDO		



En la página 60 mostramos cuatro posiciones, dos que controlan los colores de pantalla y fondo. Pruebe escribiendo esto:

POKE 53281,7 RETURN

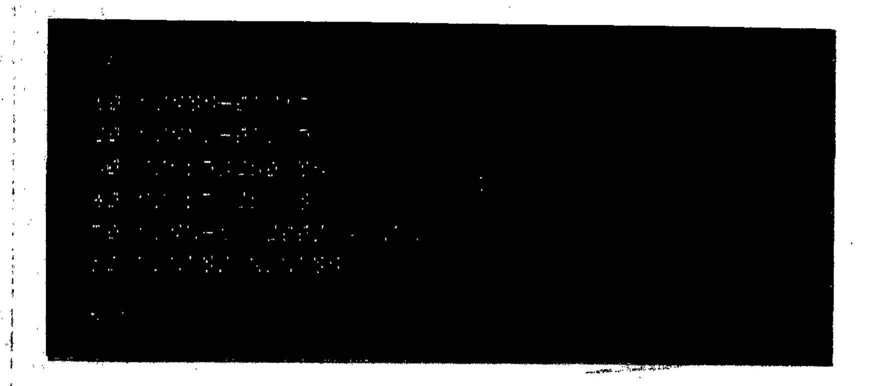
El color de fondo de la pantalla cambiará a amarillo porque ha situado el valor "7" —para amarillo— en la posición que controla el color de fondo de la pantalla.

Pruebe haciendo POKEs con diferentes valores en la posición del color de fondo, y vea qué resultados obtiene. Puede hacer POKE con cualquier valor entre 0 y 255, pero sólo de 0 a 15 trabajará.

Los valores de POKE para cada color son:

0	NEGRO	8	NARANJA
1	BLANCO	9	MARRON
2	ROJO	10	ROJO CLARO
3	CIAN	11	GRIS 1
4	PURPURA	12	GRIS 2
5	VERDE	13	VERDE CLARO
6	AZUL	14	AZUL CLARO
7	AMARILLO	15	GRIS 3

¿Puede pensar una manera de presentar varias combinaciones de bordes y fondos? Lo siguiente puede ayudarle algo:



Dos simples bucles ponen para POKE varios valores para cambiar los colores de fondo y borde. El bucle de RETARDO en la línea 50 lo retrasa un poco.



Por curiosidad pruebe:

? PEEK (53280) AND 15

Obtendrá el valor de 15. Este es el último valor BORDE que se dió y es por lo que los colores de fondo y borde son GRIS (valor 15) después de que el programa se ha ejecutado.

Entrando AND 15 elimina todo otro valor excepto 1—15, por la manera en que los códigos de color están almacenados en el ordenador. Normalmente encontrará los mismos valores que fueron POKE en la posición.

En general, PEEK nos examina una posición de memoria específica y ve que valor está presente allí. Puede pensar en añadir una línea al programa para que presente el valor de borde y de fondo cuando el programa empiece. Algo así:

```
25 PRINT CHR$(147); "BORDE = "; PEEK(53280) AND 15, "FONDO = ";
   PEEK(53281) AND 15
```

[illegible]

En toda la impresión de información que ha visto hasta ahora, el ordenador imprime toda información en una forma secuencial: un caracter es impreso después del siguiente, empezando desde la posición actual del cursor (excepto cuando pregunta por una nueva línea, o usa las "" en el formato PRINT).

Para hacer un PRINT de datos en un punto particular debe empezar por conocer la situación actual de la pantalla y hacer el PRINT con el número de controles de cursor necesarios para formatear la presentación. Pero esto toma pasos de programa y es tiempo consumido.

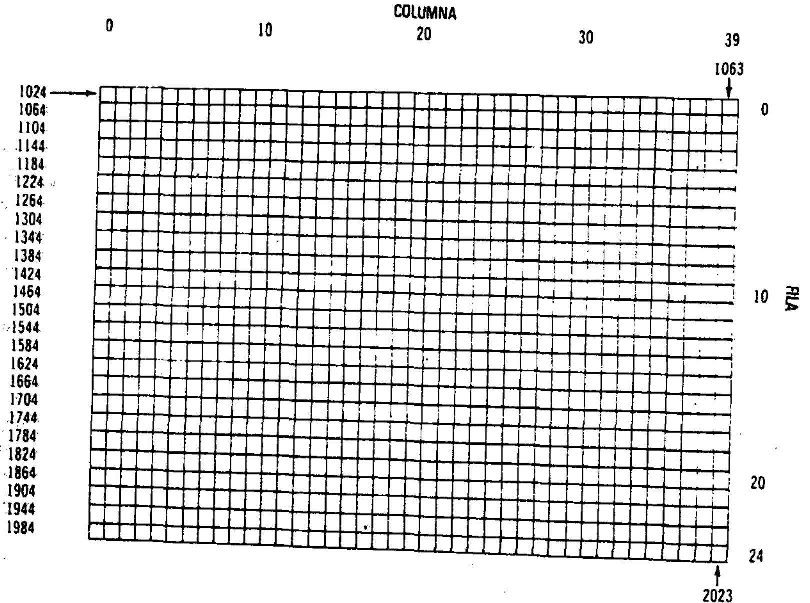
Pero como hay ciertos puntos en la memoria del Commodore 64 para el control de color, hay también posiciones que pueden usarse para controlar directamente cada posición de la pantalla.

[illegible] DE MEMORIA DE PANTALLA

Puesto que la pantalla del ordenador es capaz de contener 1000 caracteres (40 columnas por 25 líneas) hay 1000 posiciones de memoria sucesivas para manejar lo que se sitúa en la pantalla. La disposición de la pantalla puede ser como una parrilla, con cada cuadrado representando una posición de memoria.

Puesto que cada posición en memoria puede contener un número de 0 a 255, hay 256 posibles valores para cada posición en memoria. Esos valores representan los diferentes caracteres que el Commodore 64 puede presentar (ver Apéndice E). Realizando POKE con el valor de un caracter en la posición

de memoria adecuada, este caracter será presentado en esta posición.



La memoria de pantalla en el Commodore 64 normalmente empieza en la posición de memoria 1024, y finaliza en la posición 2023. La posición 1024 es la esquina superior izquierda de la pantalla. La posición 1025 es la posición del próximo caracter de la derecha de éste, y así hacia abajo de la fila. La posición 1063 es la posición más a la derecha de la primera fila. la próxima posición del último caracter en una fila es el primer caracter de la fila de abajo.

Ahora, vamos a controlar una pelota rebotando en la pantalla. La pelota está en medio de la pantalla, columna 20, fila 12. La fórmula para calcular la posición de memoria en la pantalla es:

PUNTO = 1Ø24 + X + 4Ø*Y ← X: COLUMNA, Y: FILA

donde X es la columna e Y es la fila.

Por esto, la posición de memoria de la pelota es:

1Ø24 + 2Ø + 48Ø o 1524 ← 2Ø: COLUMNA, 48Ø: FILA (40*12)

Limpie la pantalla con SHIFT y CLR HOME y entre:

POKE 1524,81 ← CODIGO DEL CARACTER
↑ POSICION DE MEMORIA

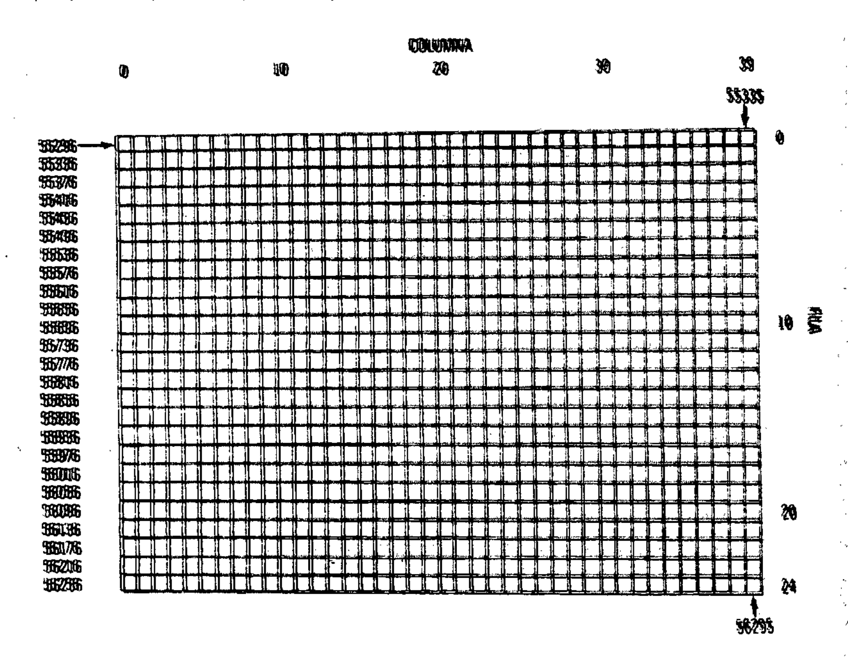
POKE 55796,1 ← CODIGO DE COLOR
↑ POSICION

MAPA DE MEMORIA DE COLOR

Ahora una pelota aparece en medio de la pantalla, Vd. ha puesto un caracter directamente en la memoria de pantalla sin utilizar la instrucción PRINT. La pelota que ha aparecido es de color blanco. No obstante existe una manera de cambiar el color de un objeto en la pantalla alterando el contenido de otro espacio de memoria, teclee:

POKE 55796,2 ← COLOR
↑ POSICION DE MEMORIA

El color de la pelota cambia al rojo. Para cada posición de caracter en la pantalla del COMMODORE 64 existen asociadas dos posiciones de memoria, una para el código del caracter y otra para el color con el que este caracter se exhibirá en la pantalla. La memoria para el color empieza en la posición 55296 (esquina superior izquierda), y continua durante 1000 posiciones. Los mismos





códigos de color, de 0 a 15, que se utilizan para cambiar el color del borde y el fondo de la pantalla sirven para definir el color de los caracteres.

La fórmula que utilizábamos para cálcular las posiciones de los caracteres en la pantalla puede modificarse para obtener los POKES donde cambiar los colores. La fórmula es:

POSICION DE COLOR = 55296 + X + 40*Y

MAS PELOTAS REBOTANDO

Aquí tiene una versión mejorada del programa que saca las figuras directamente en pantalla mediante POKE en vez de utilizar el PRINT. Como verá cuando ejecute el programa es mucho más flexible que la versión anterior y permite una animación mucho más sofisticada.

```
NEW

10 PRINT"J":REM"J ES SHIFT CLR/HOME
20 POKE53280,7:POKE53281,13
30 X=1:Y=1
40 DX=1:DY=1
50 POKE1024+X+40*Y,81
60 FORT=1TO10:NEXT
70 POKE1024+X+40*Y,32
80 X=X+DX
90 IFX=0ORX=39THENDX=-DX
100 Y=Y+DY
110 IFY=0ORY=24THENDY=-DY
120 GOTO50
```

La línea 10 limpia la pantalla, la 20 hace que el color del fondo sea el verde claro y el color del borde el amarillo.

Las variables X e Y en la línea 30 mantienen la posición actual de la pelota. DX y DY en la línea 40 tienen el valor de las direcciones horizontal y vertical del movimiento de la pelota. Cuando se suma 1 al valor de la variable X, la pelota se mueve a la derecha; cuando se le resta 1 (se le suma —1) se mueve a la izquierda. Los mismos valores en la variable Y hacen que la pelota suba o baje una fila.

La línea 50 pone la pelota en la pantalla en la posición señalada por el cursor. La línea 60 es el bucle de retardo que permite mantener la pelota en la pantalla para verla.

La línea 70 borra la pelota situando un espacio (código 32) en la posición que ocupa la pelota.

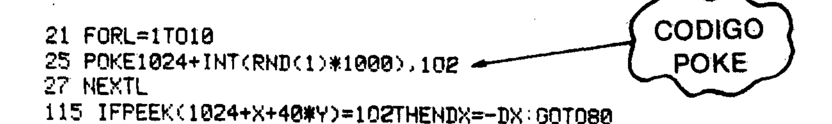


La línea 80 suma el factor a X. La línea 90 verifica si la pelota ha tocado alguna de las paredes laterales, invirtiendo la dirección si ha habido un rebote. Las líneas 100 y 110 hacen lo mismo para las paredes superior e inferior.

La línea 120 envia el programa a realizar el dibujo y efectuar el movimiento de nuevo.

Cambiando el código en la línea 50 de 81 a otro cualquiera, se puede cambiar la pelota a otro caracter. Si cambia DX o DY a 0 la pelota se moverá en perpendicular en vez de en diagonal.

Podemos también añadir ciertas mejoras. De momento la única cosa que se verifica son los valores X e Y por si se salen de la pantalla. Añada las líneas siguientes:




```
:
21 FORL=1TO10
25 POKE1024+INT(RND(1)*1000),102
27 NEXTL
115 IFPEEK(1024+X+40*Y)=102THENDX=-DX:GOTO80
```

Las líneas 21 a 27 situan 10 bloques en la pantalla en posiciones aleatorias. La línea 115 verifica, mediante un PEEK si la pelota va a rebotar contra un bloque, y cambia la dirección si es así.

INTRODUCCIÓN A LOS SPRITE

En los capítulos anteriores tratamos con gráficos, y dijimos que estos símbolos gráficos pueden usarse en sentencias PRINT para crear animación y con unas apariciones similares en su pantalla.

Una manera es mostrando los códigos de caracter POKE en posiciones de memoria de pantalla específicas.

La creación de animación en ambos casos requiere un lote de trabajo porque los objetos deben crearse a partir de los símbolos gráficos existentes. El movimiento del objeto requiere un número de instrucciones de programa para seguir la pista al objeto y moverlo a un nuevo punto. Y, por la limitación del uso de los símbolos gráficos, la forma y resolución del objeto no puede ser lo buena que se requiere.

Usando Sprite en las secuencias animadas se eliminan todos los problemas. Un Sprite es un objetivo programable en alta resolución y puede ser construido dentro, de alguna forma, por comandos BASIC. El objeto puede moverse fácilmente alrededor de la pantalla diciendo de alguna manera al ordenador la posición a la que tendría de moverse el Sprite. El ordenador hace el resto.

Y los Sprite tienen mucha más potencia que eso. Su color puede cambiarse; puede decir si uno de los objetos colisiona con otro; puede hacer ir por delante y por detrás; y se puede aumentar de tamaño fácilmente, sólo para empezar.

La dificultad de todo esto es mínima. De cualquier modo, el uso de Sprite requiere conocer ciertos detalles más acerca de como opera el Commodore 64 y como son tratados dentro del ordenador. No es tan difícil como el sonido. Siga estos ejemplos y estará construyendo sus propios Sprite haciendo cosas asombrosas en poco tiempo.



SPRITE GRÁFICOS:

Los Sprite se controlan gracias a un editor de imagen separado, en el Commodore-64. El editor de imagen se encarga de la pantalla y de la parte video. Se ocupa de todas las tareas de crear y almacenar gráficos o/y caracteres, crear colores.

El circuito de visualización tiene 46 posiciones diferentes tipo "ON/OFF" que trabajan como las situaciones de memoria interna. Cada una de éstas se compone de una serie de 8 bloques. Y cada bloque puede, a su vez estar en posición "on" o "off". Ya entraremos en más detalles un poco más tarde. Haciendo un POKE, con un debido valor decimal, en la casilla de memoria apropiada puede controlar la formación y movimiento de sus creaciones de Sprite.

Además de acceder a varias registros de memoria del editor de imagen, utilizaremos también algunos registros de la memoria general del Commodore-64 para almacenar información que definan los Sprite. Finalmente, ocho situaciones de memoria, inmediatamente consecutivas a la memoria de pantalla, se utilizarán para indicar exactamente al ordenador de que sector de la memoria cada Sprite saca sus datos.

A medida que veamos los ejemplos, el proceso será más sencillo y nos encontraremos más a gusto.

Por lo tanto, adelante, y a crear algún Sprite. Un Sprite se inscribe en una matriz de 24 x 21 puntos. Más de ocho Sprite pueden controlarse al mismo tiempo. Estos se presentan en un modo especial independiente de 320 x 200 puntos de área. Luego se pueden utilizar en Alta y Baja resolución, Texto, etc.

Pongamos que quiero crear un balón, y que quiero verlo volar por los cielos. El balón puede diseñarse en una parrilla de 21 x 24 (vea p. 70).

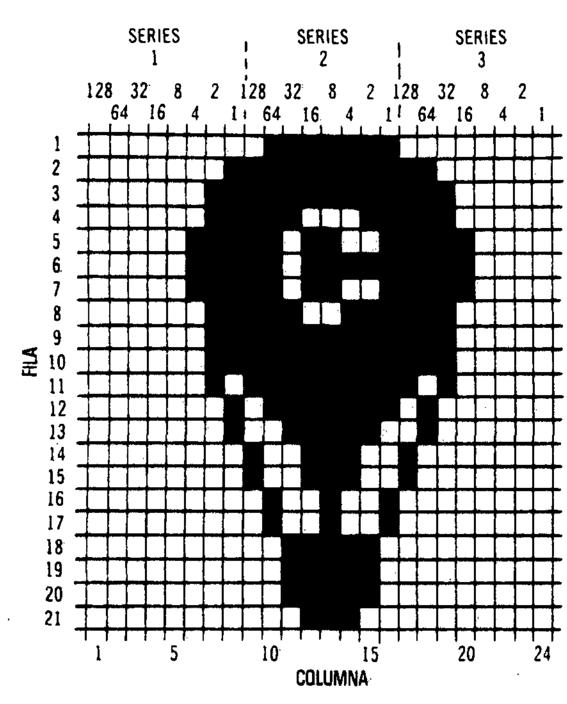
El próximo, paso será, convertir el diseño gráfico en datos asimilables por el ordenador. Coja un bloc o una hoja cuadriculada y dibuje una parrilla de 21 líneas x 24 columnas. En la parte superior horizontal marque los números 128,64,32,16,8,4,2,1, tres veces como se muestra, para cada una de las 24 casillas. Numere verticalmente las casillas de 1 a 21.

Escriba la palabra DATA al final de cada línea. Ahora rellene la parrilla con cualquier diseño o con nuestro balón. Es más fácil esbozar los contornos y luego rellenar las casillas.

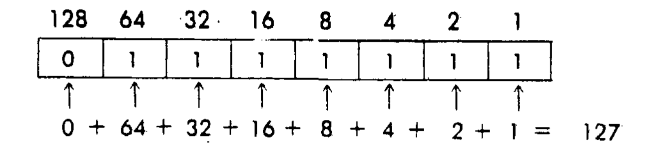
Ahora, remplaze por un uno todas las casillas que están rellenas (o en estado "on") para las que están vacías (estado "off") ponga un Ø (o dejar en blanco).

Empezando por la primera línea, debe convertir los puntos en tres unidades de datos que sean comprensibles al ordenador. Cada serie de ocho casillas es igual a una unidad de datos llamada byte. Trabajando desde la izquierda, las ocho primeras casillas estan libres, o en Ø, por lo tanto el valor para esta serie es Ø.





Las series del medio tienen este aspecto (de nuevo el 1 indica un punto y 0 un espacio):

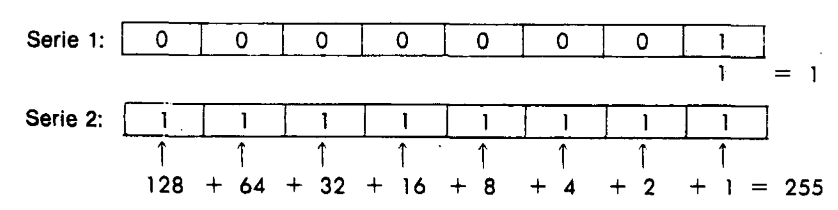


128	64	32	16	8	4	2	1	
0	1	1	1	1	1	1	1	
↑	↑	↑	↑	↑	↑	↑	↑	
0 +	64 +	32 +	16 +	8 +	4 +	2 +	1 =	127

Las series del tercer grupo de la primera línea también contienen sólo 0, por lo tanto el valor de la serie es 0. Así pués el dato para la primera línea es:

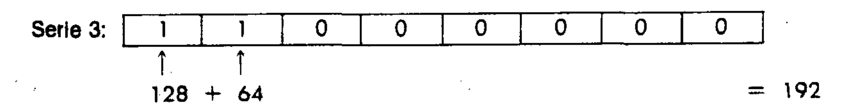
DATA Ø, 127,Ø

La serie de la línea siguiente se calculan así:



Serie 1:	0	0	0	0	0	0	0	1	
								1	= 1
Serie 2:	1	1	1	1	1	1	1	1	
	↑	↑	↑	↑	↑	↑	↑	↑	
	128 +	64 +	32 +	16 +	8 +	4 +	2 +	1	= 255





Serie 3:	1	1	0	0	0	0	0	0	
	↑	↑							
	128 +	64							= 192

Para la 2.ª línea el DATA será:

DATA 1, 255, 192

De la misma manera, las líneas restantes, se convierten en su valor decimal correspondiente. Tome el tiempo de convertir las líneas siguientes del ejemplo.

Ahora que tiene los datos de su objeto, como utilizarlos: Mecanografíe el siguiente programa y vea lo que ocurre:

```
1 REM BAJA, BAJA Y ALEJATE
5 PRINT"J"
10 V=53248:REM PUESTA EN MARCHA DEL CHIP DE PANTALLA
11 POKEV+21,4:REM PERMITE SPRITE 2
12 POKE2042,13:REM DATOS DEL PUNTO NO. 13 SPRITE 2
20 FORN=0TO62:READQ:POKE832+N,Q:NEXT
30 FORX=0TO200          <- SACA LA INFORMACION DE LOS DATA*
40 POKEV+4,X:REM DATO SOBRE COORDENADA X
50 POKEV+5,X:REM DATO SOBRE COORDENADA Y
60 NEXTX
70 GOTO30               <- INFORMA EL READ, MEDIANTE Q*
200 DATA0,127,0,1,255,192,3,255,224,3,231,224
210 DATA7,217,240,7,223,240,7,217,240,3,231,224
220 DATA3,255,224,3,255,224,2,255,160,1,127,64
230 DATA1,62,64,0,156,128,0,156,128,0,73,0,0,73,0
240 DATA0,62,0,0,62,0,0,62,0,0,28,0
```

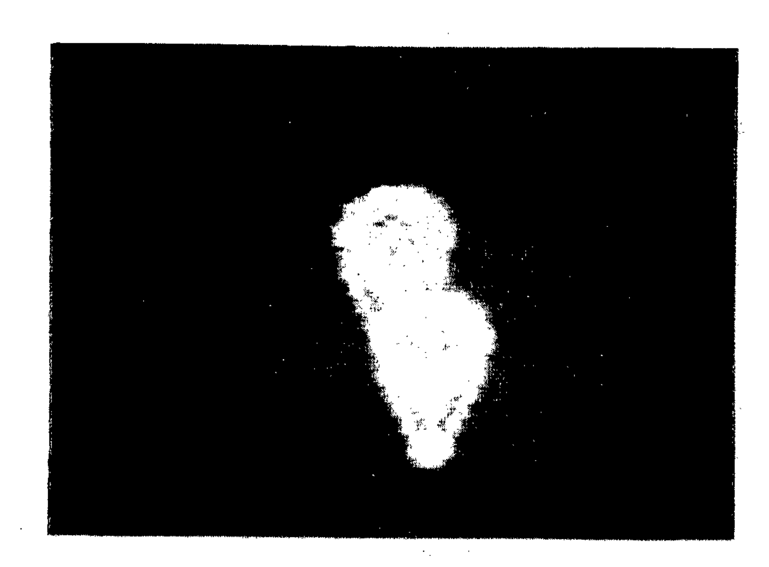
*PARA MAS DETALLES SOBRE LAS INSTRUCCIONES READ Y DATA VEA EL CAPITULO 8.

Si mecanografría todo correctamente, su balón tiene que estar volando suavemente por los cielos.

Para entender lo que ocurre, debe saber antes que registros del editor de imagen controlan las funciones que necesita. Estos registros pueden representarse como:

Registros	*Descripción*
Ø	Coordenada X o abscisa del Sprite 0
1	" Y o ordenada del Sprite 0
2 — 15	Emparejadas como 0 y 1 para los Sprite de la 7
16	Bit de mayor significado — Coordenada X.
21	Aparición del Sprite 1 = aparecido; 0 = desaparecido
29	Expansión del Sprite en dirección "X"
23	" " " " "Y"
39 — 46	Color de los Sprite 0 a 7

microelectronica y control s a



Además de esta información tiene que saber de cual de las 64 secciones de bytes (1 Sprite entero) consta cada serie de 8 registros de memoria (1 línea de Sprite haciendo DATA) dará cada Sprite su DATA (1 serie no se utiliza).

Estos datos los llevan los ocho registros siguientes a la memoria de pantalla:

	2040	41	42	43	44	45	46	2047
	↑	↑	↑	↑	↑	↑	↑	↑
Sprite.	0	1	2	3	4	5	6	7

Ahora vamos a perfilar el procedimiento exacto para mover y escribir un programa.

Sólo hay realmente unas pocas cosas que saber para crear o mover un objeto.

1. Haga aparecer el Sprite en la pantalla por POKEs en el registro 21 a 1 para que el bit del Sprite se ponga en "on".
2. Ponga el puntero de Sprite (Registro 2040-7) donde vayan a ser leídos por el ordenador.
3. POKE los datos actuales en memoria.
4. Actualice las coordenadas (X e Y), vía un bucle, para mover el Sprite
5. A título facultativo puede ampliar el objeto, cambiar colores, efectuar funciones especiales. Utilice el registro 29 para ampliar su Sprite en la dirección "X" y el registro 23 para la dirección "Y".



Sólo hay algunas pocas cosillas en el programa que no se han visto hasta ahora.

En la línea 10;

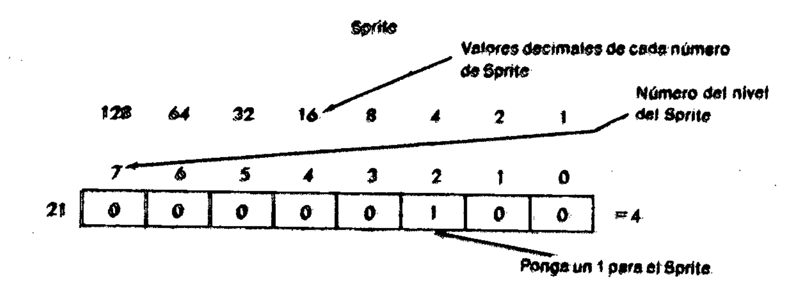
V = 53248

ajuste V al principio de las situaciones de memoria del chip de video. De esta manera se puede incrementar V, de un número de memoria para obtener el emplazamiento de memoria actual. Los números de memoria son los que da el mapa de registros.

En la línea 11;

POKE V + 21,4

hace que el Sprite 2 aparezca, con un 4 en lo que se llama el registro de disponibilidad (21) para lanzar el Sprite 2. Concebido de esta manera:



Cada nivel del Sprite se representa en la sección 21 de la memoria de Sprite y el 4 resulta ser el nivel de Sprite 2. Si utilizase el nivel 3 pondrá un 1 en el Sprite 3 y tendrá un valor de 8. Y si de hecho utiliza ambos (el 2 y el 3) pondría un 1 en 4 y 8. Luego sumaría los números, de la misma manera que lo hizo con los DATA de sus propios Sprite

Por lo tanto poniendo en marcha los Sprite 2 y 3 la cosa sería representada así V + 21,12.

En la línea 12;

POKE 2042,13

Esta instrucción indica al ordenador, de sacar datos sobre el Sprite 2 (memoria 2042, del 13avo registro de memorial. Sabe, por haber creado su propio Sprite, que ocupa 63 secciones de memoria. A lo mejor no se ha dado cuenta que los números que se ponen arriba de la primera línea de su parrilla, no son otra cosa que 3 bytes de memoria. Dicho de otra forma, cada colección de los números siguientes 128,64,32,16,8,4,2,1 equivale a un byte de memoria. Por eso son las 21 líneas de su parrilla, a razón de 3 bytes por línea, cada Sprite necesita más de 63 bytes de memoria.

SPRITE ENTERO

```
2∅ FOR N = ∅ to 62: READ Q: POKE 832 + N,Q: NEXT
```




Esta es la línea clave que se ocupa realmente del Sprite creado. Los 63 bytes de datos que representan al Sprite que creó, son leídos (READ) por el bucle y POKE puestos en el 13avo bloque de memoria, y esto empieza en la situación 832.

```
30 FORX=0TO200
40 POKEV+4,X    (COORDENADAS X DEL SPRITE 2)
50 POKEV+5,X    (COORDENADAS Y DEL SPRITE 2)
```

Si recuerda un poco la escuela, se acordará que la coordenada X o abscisa, se presenta en posición horizontal, y la coordenada Y la situación vertical del Sprite en la pantalla. Por eso como los valores de X cambian en la línea 30 de 0 a 200 (un número a la vez), el Sprite se mueve **através** de la pantalla ABAJO y a la DERECHA, en espacio por número. Los números son leídos por el ordenador, suficientemente rápido para que aparezca el movimiento como contínuo, en vez de 1 por 1. Si necesita más detalles eche un vistazo al mapa de resgistros en el apéndice.

Cuando llegue a mover varios objetos en la pantalla, sería imposible a una sóla sección de memoria el actualizar las situaciones de los 8 objetos. Por eso, cada Sprite tiene su propia serie de 2 secciones de memoria para hacerlo mover sobre la pantalla.

La línea 70 empieza el ciclo de nuevo, después de una pasada por la pantalla **(lo que recuerda al programa el balón son los DATA).** Verdad que es muy diferente en la pantalla!

Ahora pruebe añadiendo la línea siguiente.

```
                 AMPLIA X               AMPLIA Y
25 POKE V+23,4 : POKE V+29,4 : REM: EXPANSION / AMPLIACION
```

y lance de nuevo el programa. El balón se ha ampliado de dos veces su tamaño original. Lo que hicimos es sencillo. POKE (poniendo) 4 (para indicar, de nuevo, el Sprite 2) en el registro 23 y 29, el Sprite 2 se amplió en las direcciones X e Y.

Es importante observar que el origen de un Sprite está en el esquina izquierda superior del objeto. Cuando amplie un objeto en las dos dimensiones, el origen no cambia. Para más diversión, añada esto:

```
11 POKEV+21,12
12 POKE2042,13:POKE2043,13
30 FORX=1TO190
45 POKEV+6,X
55 POKEV+7,190-X
```

Un segundo Sprite (el n.º 3) se ha dispuesto POKEando 12 en la situación de memoria que hace aparecer un Sprite (V+21). El "12" transforma en bits el 2 y 3 (00001100 = 12).



Las líneas 45 y 55, que hemos añadido, desplazan el Sprite 3 haciendo un POKE en las situaciones de memoria de las coordenadas X e Y del Sprite 3 (V + 6 y V + 7).

Quiere más acción.

```
11 POKEV+21,28    (Entonces añada las líneas siguientes.)
12 POKE2042,13:POKE2043,13:POKE2044,13
25 POKEV+23,12:POKEV+29,12
48 POKEV+8,X
58 POKEV+9,100
```

En la línea 11, esta vez, otro Sprite, el 4, aparece, introduciendo por POKE 28, en la apropiada situación "on" de la sección de memoria de Sprite. Ahora los sprites 2-4 están en "on" (00011100 = 28).

La línea 12 indica que el Sprite 4 sacará los datos de la misma área de memoria **(13 ava área de 63 secciones)** como los otros Sprite con POKE 2044,13.

En la línea 25, los Sprite 2 y 3 se amplían introduciendo por POKE, 12 (Sprite 2 y 3 en las situaciones de memoria, de expansión de dirección de las coordenadas X e Y (V+23 y V+29).

La línea 48 desplaza el Sprite 4 a lo largo **del eje X.** La línea 58 situa el Sprite 4, en la mitad (vertical) de la pantalla, en la situación 100.

Puesto que este valor no cambia, como anteriormente con X=0 a 200, el Sprite 4 sólo se mueve horizontalmente.

NOTAS ADICIONALES SOBRE LOS SPRITE

Ahora que ya conoce los Sprite y ha experimentado con ellos, unas últimas palabras son adecuadas. Primero se puede cambiar el color de un Sprite a cualquiera de los 16 colores estandards, que utilizamos para cambiar los colores de los caracteres. Puede encontrar información sobre ello en el capítulo 5 y en el apéndice G.

Por ejemplo, para cambiar el Sprite 1 a verde claro, escriba POKE V + 40,13 (asegurese de haber establecido V = 53248).

Habrá notado, utilizando el ejemplo de programa de Sprite, que éste nunca se desplaza hasta el borde derecho de la pantalla. Esto es debido a que la pantalla es de 320 puntos de largo y el registro de dirección X sólo puede llevar un valor de hasta 255. Entonces ¿cómo hacer desplazar un objeto a lo largo de toda la pantalla?

Hay una situación en el mapa de memoria de la cuál no hemos hablado aún. La situación 16 del mapa controla algo llamado el bit más significativo B.M.S. (M.S.B. en Inglés) del registro de dirección del Sprite En efecto, ésto le permite mover el Sprite horizontalmente de la posición 255 a 320.



El B.M.S. del registro X trabaja así: una vez que el Sprite está en la posición 255, pone un valor en la situación de memoria 16 representando así, el Sprite que Ud. quiere mover. Por ejemplo, si quiere que el n.° 2 se desplace a las posiciones horizontales 256-320, haga un POKE para el Sprite n.° 2, es decir el valor 4, en el registro 16.

POKE V + 16,4

Ahora, empieze desde Ø de nuevo en la dirección del registro usual del Sprite 2 (cuando está en la posición 4 del mapa) Cómo sólo está moviendo otros 64 espacios, el registro X sólo va de Ø a 63.

Todo este concepto se ilustra mucho mejor con una nueva versión del programa original del Sprite 1:

```
10 V=53248:POKEV+21,4:POKE2042,13
20 FORN=0TO62:READQ:POKE832+N,Q:NEXT
25 POKEV+5,100
30 FORX=0TO255
40 POKEV+4,X
50 NEXT
60 POKEV+16,4
70 FORX=0TO63
80 POKEV+4,X
90 NEXT
100 POKEV+16,0
110 GOTO30
```

La línea 6Ø ajusta el B.M.S. para el Sprite n.° 2. La línea 7Ø empieza a mover el registro de dirección X, desplazando así el Sprite n.° 2 hasta el borde de la pantalla.

La línea 1ØØ es importante puesto que pone a cero el B.M.S. de manera que el Sprite pueda moverse de nuevo a partir del borde izquierdo de la pantalla. Para definir Sprite múltiples, necesitará bloques adicionales para los datos de los Sprite Puede usar la RAM de BASIC moviendo el BASIC. Antes escriba y cargue este programa.

POKE 44,16 : POKE 16*256,Ø : NEW

Ahora, puede usar los bloques 32 a 41 (posiciones 2048 a 4095) para almacenar los datos de los Sprite

LA ARITMETICA BINARIA

Está fuera de los objetivos de este manual el mostrarle todos los detalles sobre como el ordenador manipula los números. Sin embargo, le ofrecemos todo lo necesario para que tenga buenas bases y pueda comprender el proceso e iniciarse a la animación sofisticada.

Pero antes de entrar en los detalles, vamos a definir algunos términos:



***BIT:** es la menor cantidad de información que puede almacenar el ordenador. Piense en un bit como un interruptor que puede estar en la posición encendido (ON) o apagado (OFF). Cuando está en la posición "ON", tiene el valor 1, si está en posición "OFF" el valor 0.

Después del BIT el nivel siguiente es el BYTE.

***BYTE:** Se define como una serie de bits. Visto que un byte se compone de 8 bits se pueden obtener así hasta 256 combinaciones diferentes de bits. En otras palabras su BYTE puede tener todos los BITS en posición "OFF" y tendrá este aspecto:

128	64	32	16	8	4	2	1
0	0	0	0	0	0	0	0

o tener todos los bits en posición "ON" y tener esta forma:

128	64	32	16	8	4	2	1
1	1	1	1	1	1	1	1

lo que da 128 + 64 + 32 + 16 + 8 + 2 + 1 = 255.

Después viene el REGISTRO.

***REGISTRO:** Definido como un bloque de BYTES "ensamblados", en este caso cada registro es de 1 sólo byte. Una serie de registros forman un MAPA de REGISTROS. Los MAPAS de registros están estructurados como los vió cuando creamos el balón. Cada registro controla una función diferente, como por ejemplo el registro de puesta en marcha de un Sprite Cuando se amplía un Sprite, se utilizán los REGISTROS DE EXPANSION X e Y. Acuérdese sencillamente de que un REGISTRO es un byte que efectúa una tarea especifica.

Ahora veamos otros aspectos de la ARITME TICA BINARIA.

CONVERSION DE NUMEROS BINARIOS A DECIMALES:

Valor Decimal								
128	64	32	16	8	4	2	1	
0	0	0	0	0	0	0	1	2↑0
0	0	0	0	0	0	1	0	2↑1
0	0	0	0	0	1	0	0	2↑2
0	0	0	0	1	0	0	0	2↑3
0	0	0	1	0	0	0	0	2↑4
0	0	1	0	0	0	0	0	2↑5
0	1	0	0	0	0	0	0	2↑6
1	0	0	0	0	0	0	0	2↑7



Utilizando combinaciones de ocho bits, puede obtener cualquier valor decimal de 0 a 255. Empieza a ver por qué cuando hacemos POKE para generar un *carácter o color los valores que ponemos en las situaciones de memoria*, están comprendidos entre 0 y 255. Cada situación de memoria puede contener un byte de información.

Cualquier combinación posible de 8 "ceros" o "unos" se ha convertido en un único color decimal comprendido entre 0 y 255. Si todos los bits contienen un 1 el valor del byte será 255, si sólo tiene 0, entonces el valor sera nulo. "00000011" equivale a 3, y así... Esto será la base para crear datos que representen Sprite y manipularlos. Como ejemplo, si este byte fuese parte de un Sprite (Ø para un espacio y 1 para una área ocupada).

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
1	1	1	1	1	1	1	1	
128 +	64 +	32 +	16 +	8 +	4 +	2 +	1 +	= 255

Entonces haríamos un POKE, de valor 255 en la situación de memoria apropiada que representa esta parte del objeto.

AVISO:

Para evitarle tener que convertir números binarios a valores decimales, —y tendremos que hacerlo muy a menudo— el programa siguiente hará el trabajo. Sería una buena idea guardarlo para un uso futuro.

```
5 REM CONVERSOR BINARIO A DECIMAL
10 INPUT"ENTRE UN NUMERO BINARIO DE 8 BITS :";A$
12 IFLEN(A$)<>8THENPRINT"8 BITS POR FAVOR...":GOTO10
15 TL=0:C=0
20 FORX=8TO1STEP-1:C=C+1
30 TL=TL+VAL(MID$(A$,C,1))*2↑(X-1)
40 NEXTX
50 PRINTA$;" BINARIO ";" = ";TL;" DECIMAL"
60 GOTO10
```

Este programa, toma el número binario, que se entró en cadena y mirá cada caracter de la cadena de izquierda a derecha (función MID$). La variable C indica en que bit está trabajando a medida que el programa pasa por el bucle.

La fusión VAL, en la línea 30, afecta al carácter su valor **actual**. Como estamos trabajando con caracteres de A$ es 1, el valor será también 1.

La última parte de la linea 30, multiplica el valor del carácter en curso por su propia potencia de 2. Como el primer valor está en la casilla 2^7 ó 2→7, en el ejemplo TL se dará el valor 128. Y si el valor del bit fuese 0, el valor de la casilla también sería cero.

Este proceso se repite para los 8 bits a medida que TL guarda el valor decimal de número binario.

CAPITULO 7

CREACIÓN DE SONIDOS

- [illegible]
- [illegible]
- [illegible]
- [illegible]
- [illegible]
- Creación Efectos Especiales

USO DEL SONIDO SI NO ES UD. UN PROGRAMADOR

La mayor parte de los programadores utilizan el sonido para dos propósitos: la creación musical y los efectos especiales. Pero antes de entrar en el intrincado mundo de la programación de los sonidos, echemos un vistazo rápido a la forma en que está estructurado un programa musical. Además le vamos a dar un pequeño programa con el que pueda experimentar.

ESTRUCTURA DE UN PROGRAMA DE SONIDO

Para empezar hay cinco ajustes que deberá conocer para generar sonidos con su **COMMODORE-64: EL VOLUMEN, ATAQUE/DECAIMIENTO, SOSTENIMIENTO/RELAJACION (ADSR), CONTROL DE ENVOLVENTE y ALTA/BAJA FRECUENCIA.** Las cuatro primeras funciones es usual ponerlas UNA VEZ en el principio del programa. El ajuste de la alta y baja frecuencia se hace para cada nota que se toca. Este es un ejemplo de cómo está ESTRUCTURADO un programa musical.

EJEMPLO DE PROGRAMA MUSICAL

Antes de empezar debe escoger una **VOZ.** Hay tres voces. Cada voz requiere un ajuste de sonido diferente para la envolvente, etc... Puede tocar una, dos o tres simultáneamente, pero nuestro ejemplo sólo utiliza la voz número 1. Escriba este programa línea por línea. No olvide pulsar RETURN al final de cada línea:

1. Primero limpie el chip de sonido	**5 FORL = 54272 TO 54295:POKEL,0:NEXT**
2. Ajuste el volumen al máximo	**10 POKE 54296,15**
3. Ajuste de los niveles de **ATAQUE DECAIMIENTO** para definir a que velocidad una nota sube y cae desde su nivel de volumen pico (0 a 255).	**20 POKE 54277,190**
4. Ajuste del nivel **SOSTENIMIENTO RELAJACION** para prolongar la nota a un cierto volumen y soltarla proporcionalmente.	**30 POKE 54278,248**
4. Busque la nota y tono que quiere tocar en la TABLA DE NOTAS MUSICALES del Apéndice M y entre los valores de Alta y Baja frecuencia para esa nota (Cada nota requiere dos POKEs).	**40 POKE 54273,17:POKE 54272,37**

5. Ponga la **ENVOLVENTE** con uno de los cuatro ajustes estandard: (17, 33, 65 ó 129).	**50 POKE 54276,17**
6. Entre un bucle de **TIEMPO** para la duración de la nota (un cuarto de nota es aproximadamente "250", pero puede variar con la longitud del programa puesto que ello afecta al tiempo de ejecución.	**60 FORT = 1TO250:NEXT**
7. **APAGA** la nota.	**70 POKE 54276,16**

Para oír la nota que acaba de crear, escriba RUN y pulse RETURN. Para ver o rectificar el programa escriba LIST y pulse RETURN.

MUSICA CON EL COMMODORE-64

¡No necesita ser músico para tocar algo con su COMMODORE-64! Lo único que necesita saber son unos pocos números que indican al ordenador a qué nivel poner el volumen, qué notas tocar, cuanto tiempo tocarlas, etc... Pero antes...hay un programa que le muestra rápidamente las increíbles posibilidades musicales de su COMMODORE 64, utilizando una sola voz de las tres existentes.

Escriba la palabra NEW y pulse la tecla RETURN para borrar cualquier información anterior; luego entre este programa, escriba RUN y pulse RETURN.



```
5 REM ESCALA MUSICAL
7 FORL=54272TO54296:POKEL,0:NEXT
10 POKE54296,15
20 POKE54277,7:POKE54278,133
30 POKE54276,17
40 FORT=1TO300:NEXT
50 READA
60 READB
70 IFB=-1THENPOKE54273,0:POKE54272,0:END
80 POKE54273,A:POKE54272,B

85 POKE54276,17
90 FORT=1TO250:NEXT:POKE54276,16
100 GOTO20
110 DATA17,37,19,63,21,154,22,227
120 DATA25,177,28,214,32,94,34,175
900 DATA-1,-1
```

Título del programa
Borra los registros del chip.
Ajusta el volumen al máximo.
Ajusta A/D y S/R.
Determina la forma de onda.
Duración de cada nota.
Lee el primer número en 110.
Lee el segundo número en 110.
Si lee -1 apaga el control de envolvente y acaba el programa.
Pone el primer número leído en las líneas DATAs como ALTA FRECUENCIA y el segundo número como BAJA FRECUENCIA y así sucesivamente.
Apaga control de envolvente.
Vuelve para tocar una nueva nota.
Datos musicales.
Para más información, vea el Apéndice M.

Para cambiar el sonido a un clavicordio, cambie la línea 30 por:

POKE 54276,33 y la línea 90 por: **FORT = 1TO250:NEXT:POKE54276,32**

y ejecute el programa de nuevo. (para cambiar la línea, pulse la tecla RUN/STOP para parar el programa, escriba la palabra LIST y pulse RETURN . Luego escriba de nuevo la línea de programa que quiera cambiar, la nueva reemplazará automáticamente a la antigua). Lo que hacemos aquí es cambiar la envolvente triangular a la envolvente de diente de sierra. Cambiando la envolvente, puede cambiar drásticamente el sonido producido por el COMMODORE 64... pero... ¡la envolvente es sólo uno de los múltiples controles que puede ajustar para crear diferentes notas musicales y efectos especiales! También puede cambiar la relación ATAQUE/DECAIMIENTO para cada nota... por ejemplo, para cambiar un sonido estilo clavicordio a estilo banjo, modifique la línea 20 como sigue:

20 POKE 54277,3:POKE 54278,0



Como habrá visto, puede hacer de su COMMODORE-64, el instrumento musical que más desee. Pero veamos con más detalle como funciona cada control musical.

LOS AJUSTES DE SONIDO BASICOS

1. EL VOLUMEN: Para poner en marcha el volumen y ajustarlo al máximo nivel, escriba: POKE 54296,15. Los valores del nivel de volumen van de 0 a 15, pero el que más utilizará es el nivel 15. Así pués escriba para poner el volumen:

POKE 54296,15.

Sólo tiene que ajustar UNA SOLA VEZ el volumen al principio del programa, puesto que este activa las 3 VOCES de su Commodore-64. (Cambiar el volumen durante la ejecución de una nota musical puede ser muy interesante, pero sobrepasa los objetivos de esta introducción).

2. EL ADSR y LOS CONTROLES DE FORMA DE ONDA: Ya ha visto como el modificar la ENVOLVENTE puede cambiar la sonoridad del xilofón al clavicordio. Cada voz tiene sus propios controles de FORMA DE ONDA, que le permiten definir 4 tipos diferentes de formas de onda: Triangular, Diente de sierra, Cuadrada y ruido. El CONTROL también activa al ADSR del COMMODORE-64, pero volveremos a ello dentro de un momento. Un ejemplo de puesta en marcha ajustando la ENVOLVENTE, se presenta así.

POKE 54276,17

donde el primer número (54276) representa el ajuste del control para la VOZ 1 y el segundo (17) representa a la ENVOLVENTE triangular. Las combinaciones de ajustes para cada VOZ y ENVOLVENTE se muestran en la tabla siguiente:

AJUSTE DE CONTROL DE ENVOLVENTE Y ADSR

	AJUSTE DE CONTROL	TRIANGULO	DIENTE SIERRA	CUADRADA	RUIDO
VOZ 1	54276	17	33	65	129
VOZ 2	54283	17	33	65	129
VOZ 3	54290	17	33	65	129



Aunque los controles sean diferentes para cada voz, el código de ajuste de envolvente es el mismo para todas ellas. Para ver como funciona esto, mire las líneas 30 y 90 en el programa de la escala musical. En este programa ajustamos el CONTROL de la VOZ 1 en la línea 30 escribiendo POKE 54276,17. Se pone en marcha el CONTROL de la VOZ 1 y ajusta la envolvente triangular (17). En la línea 70 escribimos POKE 54276,0, parando la nota. Más adelante, cambiamos el principio de envolvente cambiando de 17 a 33 creando la ONDA en DIENTE DE SIERRA y ésta da a la escala el efecto de clavicordio. Veamos ahora qué interacción hay entre el CONTROL y el AJUSTE DE ENVOLVENTE. Ajustar la envolvente es similar a ajustar el VOLUMEN, excepto que cada voz tiene un propio ajuste y que en vez de hacer un POKE de un código de nivel de volumen es un código de envolvente. Pronto veremos otro aspecto del sonido... el ADSR.

3. EL AJUSTE DEL ATAQUE/DECAIMIENTO. Como mencionamos anteriormente, el ajuste del CONTROL ADSR no sólo define la envolvente sino que, además activa el **ADSR** o **ATAQUE/DECAIMIENTO/SOSTENIMIENTO/RELAJACION**, otra fantástica característica del COMMODORE-64. Empezaremos por ver el ajuste de ATAQUE/DECAIMIENTO. La tabla que sigue le mostrará los diversos niveles de ATAQUE/DECAIMIENTO para cada voz. Si no está familiarizado con los conceptos musicales de ATAQUE y DECAIMIENTO, sólo tiene que pensar en el ATAQUE como una *relación* que hace que una nota/sonido suba a su máximo nivel. El DECAIMIENTO es la *relación* con que una nota sonido cae desde su nivel máximo de volumen apoyándose en el nivel SOSTENIMIENTO. La tabla siguiente muestra los ajustes de ATAQUE/DECAIMIENTO para cada voz y sus *códigos*. Observe que DEBE COMBINAR EL ATAQUE Y EL DECAIMIENTO, SUMANDO LOS CODIGOS Y ENTRANDO EL TOTAL. Por ejemplo, puede ajustar un ATAQUE FUERTE y un DECAIMIENTO SUAVE añadiendo el código de Ataque *fuerte* (64) al *decaimiento suave* (1). El total (65) indicará al ordenador que *ataque fuerte* y *que haga una caída suave*. También puede incrementar las relaciones de ataque sumándolas (128 + 64 + 32 + 16 = 240 max. velocidad de ataque).

AJUSTE DE LA RELACION ATAQUE/DECAIMIENTO

A/D		ATAQUE				DECAIMIENTO			
		FUERTE	MEDIO	BAJO	SUAVE	FUERTE	MEDIO	BAJO	SUAVE
VOZ 1	54277	128	64	32	16	8	4	2	1
VOZ 2	54284	128	64	32	16	8	4	2	1
VOZ 3	54291	128	64	32	16	8	4	2	1

Si ajusta una *relación* de ataque sin decaimiento, el decaimiento será automáticamente cero y viceversa. Por ejemplo si hace un POKE 54277,64 está usted ajustando un ataque medio con una relación de decaimiento nula, para la VOZ 1. Si hace el POKE 54277,66, está ajustando un ataque medio, y una caída baja (puesto que 66 = 64 + 2, y esto activa ambos ajustes). También puede sumar varios valores de ataque o de decaimiento. Por ejemplo, puede tomar un



ataque bajo (32) y un ataque medio (64) para obtener un total de 96 y combinarle otro decaimiento de 4, lo que le da... POKE 54277,100.

En estos momentos lo que ilustra mejor estos efectos es un programa. Escriba NEW, pulse RETURN, entre y ejecute el siguiente programa:

```
5 FORL=54272TO54296:POKEL,0:NEXT —— Limpia los registros del chip
10 PRINT"PULSE UNA TECLA" —— Mensaje de pantalla
20 POKE54296,15 —— Ajusta volumen al máximo
30 POKE54277,64 —— Ajusta A/D
40 POKE54273,17:POKE54272,37 —— POKE una nota (VOZ 1)
60 GETK$:IFK$=""THEN60 —— Espera una tecla
70 POKE54276,17:FORT=1TO200:NEXT —— Ajuste envolvente (triángulo)
80 POKE54276,16:FORT=1TO50:NEXT —— Desactiva envolvente triangular
90 GOTO60 —— Repite de nuevo la nota
```

Aquí, estamos utilizando la VOZ 1 para crear una sola nota a la vez... con una relación de ataque medio, y una decaimiento cero. La línea clave es la línea 30. Haciendo un POKE de ajuste de ATAQUE/DECAIMIENTO con el código 64, activará una relación de ataque medio. El sonido resultante se parecerá al ruido que haríamos golpeando una lata de aceite vacía. Ahora para ver la parte divertida del asunto, pulse la tecla RUN/STOP para parar la ejecución del programa, luego escriba la palabra LIST y pulse RETURN. Ahora escriba la línea siguiente y pulse RETURN (está reemplazará automáticamente la antigüa línea 30).

30 POKE 54277,190

Escriba la palabra RUN y pulse RETURN, para ver cómo suena. Lo que hemos hecho aquí, es combinar varias relaciones de Ataque/Decaimiento. Los ajustes son los siguientes: **ATAQUE FUERTE (128) + ATAQUE BAJO (32) + ATAQUE SUAVE (16) + CAIDA FUERTE (8) + DECAIMIENTO MEDIA (4) + DECAIMIENTO BAJO (2)** = 190. Esto suena a oboe. Si le gusta experimentar, pruebe el cambiar los valores de la envolvente y las relaciones de Ataque/Decaimiento en el ejemplo de escala musical para ver como suena un oboe. Así...veremos que el cambiar las relaciones ataque/decaimiento se producen diferentes tipos de efectos especiales.

4. SOSTENIMIENTO/RELAJACION. *De la misma manera que el ataque/decaimiento, el ajuste de* SOSTENIMIENTO/RELAJACION *se activa por el control* ADSR/ENVOLVENTE. *El* SOSTENIMIENTO/RELAJACION *le permite extender una parte de un sonido particular, como el pedal de* SOSTENIMIENTO *de un piano u órgano que le permite prolongar una nota. Cualquier nota o sonido se puede* SOSTENER *en cualquiera de los 16 niveles de volumen... incluso se puede sostener el sonido en su nivel máximo (240), sin relajación, para que suene una nota "indefinidamente". El ajuste de* SOSTENIMIENTO/RELAJACION *debe ser utilizado con un bucle* FOR... NEXT,



para determinar cuanto tiempo una nota se sostendrá en su nivel de volumen máximo antes de relajarse. La tabla siguiente muestra los códigos numéricos que debe entrar con POKE para tener acceso a diversas duraciones de SOSTENIMIENTO/RELAJACION.

AJUSTE DE LA RELACION DE SOSTENIMIENTO/RELAJACION

CONTROL	SOSTENIMIENTO					RELAJACION			
S/R	ALTO	MEDIO	BAJO	SUAVE		ALTA	MEDIA	BAJA	SUAVE
VOZ 1	54278	128	64	32	16	8	4	2	1
VOZ 2	54285	128	64	32	16	8	4	2	1
VOZ 3	54292	128	64	32	16	8	4	2	1

A título de ejemplo, si utiliza la VOZ 1, puede ajustar un nivel de SOSTENIMIENTO FUERTE con un POKE 54278,128 o puede combinar un nivel de SOSTENIMIENTO FUERTE con una RELAJACION BAJA sumando 128 + 2 (128 + 2 = 130) con el POKE 54278,130. Este es el mismo ejemplo que utilizamos para la relación Ataque/Decaimiento, con la posibilidad de Sostenimiento/Relajación. Note la diferencia en los sonidos:

```
5 FORL=54272TO54296:POKEL,0:NEXT — Limpia registros del chip
10 POKE54296,15 ——————— Ajusta volumen al máximo
20 POKE54277,64 ——————— Ajusta A/D
30 POKE54278,128 —————— Ajusta S/R
40 POKE54273,17:POKE54272,37 ——— POKE una nota en VOZ 1.
50 PRINT"PULSE UNA TECLA" ——— Mensaje en pantalla
60 GETK$:IFK$=""THEN60 ———— Espera una tecla
70 POKE54276,17:FORT=1TO200:NEXT Ajusta envolvente (triángulo)
80 POKE54276,16:FORT=1TO50:NEXT— Puesta a cero de envolvente
90 GOTO60 ——————————— Repite nota otra vez
```

En la línea 30 decimos al ordenador que SOSTENGA la nota en un nivel de SOSTENIMIENTO FUERTE (128 de la tabla anterior)... después de esto la nota se relaja en la línea 80. Puede variar la duración de la nota cambiando el contador en la línea 70. Para ver el efecto del uso de la función Relajación cambie la línea 30 por POKE 54278,89 (SOSTENIMIENTO = 80, RELAJACION = 9).

5. SELECCION DE VOCES Y AJUSTE DE ALTA Y BAJA FRECUENCIA: Cada nota del Commodore 64 necesita 2 POKEs diferentes...uno para BAJA FRECUENCIA y otro para ALTA FRECUENCIA. La tabla de los valores de las notas musicales del Apéndice M le muestra los POKEs que debe hacer para tocar la nota que quiera dentro de la gama de 8 octavas que tiene el COMMODORE 64. Los POKEs de Alta y Baja frecuencia son diferentes para cada voz —esto le



permite programar las 3 voces independientemente para crear música con 3 voces o efectos sonoros exóticos.

Los POKEs de Alta y Baja frecuencia, para cada voz se muestran en la tabla que sigue, que contiene también los valores para la octava media (la quinta octava).

VOZ N.º FRECUENCIA	N.º de POKE	NOTAS MUSICALES DE LA QUINTA OCTAVA													
		DO	DO #	RE	RE #	MI	FA	FA #	SOL	SOL	LA	LA #	SI	DO	DO #
VOZ 1/ALTA	54273	34	36	38	40	43	45	48	51	54	57	61	64	68	72
VOZ 2/BAJA	54272	75	85	126	200	52	198	127	97	111	172	126	188	149	169
VOZ 1/ALTA	54280	34	36	38	40	43	45	48	51	54	57	61	64	68	72
VOZ 2/BAJA	54279	75	85	126	200	52	198	127	97	111	172	126	188	149	169
VOZ 1/ALTA	54287	34	36	38	40	43	45	48	51	54	57	61	64	68	72
VOZ 2/BAJA	54286	75	85	126	200	52	198	127	97	111	172	126	188	149	169

Como puede ver hay dos ajustes para cada voz, uno de ALTA FRECUENCIA y otro de BAJA FRECUENCIA. Para tocar una nota musical debe poner por POKE un valor en el REGISTRO DE ALTA FRECUENCIA, y otro en el de BAJA FRECUENCIA. Utilizando los apuntes de nuestra tabla de valores de VOZ/FRECUENCIA/NOTA, estos son los ajustes que tocan un DO de la quinta octava (VOZ 1).

POKE 54273,34:POKE 54272,75.

La misma nota en la VOZ 2 sería:

POKE 54280,34:POKE 54279,75.

Utilizando en un programa, la cosa se presenta así:

```
5 FORL=54272TO54296:POKEL,0:NEXT              Limpia chip sonido
10 V=54296:W=54276:A=54277:S=54278:H=54273:L=54272
20 POKEV,15:POKEA,190:POKES,89                Volumen y ADSR
30 POKEH,34:POKEL,75                          Notas
40 POKEW,33:FORT=1TO200:NEXT                  Envolvente, duración nota
50 POKEW,32                                   Fin nota
```




TOQUE UNA CANCION CON EL COMMODORE 64

El programa siguiente se puede utilizar para componer o tocar una canción (utilizando la VOZ 1). Hay 2 lecciones importantes en este programa. Primero, observe como hemos abreviado todos los largos ajustes de controles, escribiendo en la primera línea, variables para los números de los registros de memoria...así utilizamos M para Modulación en vez de 54276.

La segundos enseñanza, concierne la manera de utilizar los DATA. Este programa está preparado para poderle entrar 3 valores para cada nota. El valor de ALTA FRECUENCIA, el de BAJA FRECUENCIA, y la DURACION de la NOTA.

Para esta canción utilizamos un contador de duración así: 125 para un 1/8 de nota; 250 para 1/4 de nota, 375 para 1/4 de nota punteado, 500 para 1/2 nota y 1000 para una nota entera. Estos valores (numéricos) pueden ser incrementados o decrementados para darle un tiempo determinado, a su gusto.

Para ver como se entra una composición, mire la línea 100. Entonces 34 y 75 como ajustes de Alta y Baja frecuencia para tocar un "DO" (del ejemplo previo) y luego el número 250 para un 1/4 de nota. Por lo tanto la primera nota que se tocará es un DO de 1/4 de nota. La segunda nota también es una negra pero en MI y así hasta el final de la canción. Puede entrar así, casi cualquier canción, añadiendo tantos DATA como sea necesario. Puede continuar, poniendo valores de notas y duración de una línea a otra, pero esta deberá estar encabezada por la palabra DATA. DATA-1,-1,-1, sería la última línea del programa. Esta línea da por finalizada la canción.

Mecanografíe la palabra NEW para borrar su programa anterior y entre el programa siguiente, luego láncelo y escuche la canción...

Título de la canción:

MICHELE ROW THE BOAT ASMORE-1: COMPAS

```
 2 FORL=54272TO54296:POKEL,Ø:NEXT
 5 V=54296:M=54276:A=54277:HF=54273:LF=54272:S=54278
   PH=54275:PL=54274
10 POKEV,15:POKEA,88:POKEPH,15:POKEPL,15:POKES,89
20 READH:IFH=-1THENEND
30 READL
40 READD
50 POKEHF,H:POKELF,L:POKEM,65
80 FORT=1TOD:NEXT:POKEM,64
85 FORT=1TO50:NEXT
```





```
90 GOTO1Ø
100 DATA34,75,25Ø,43,52,25Ø,51,97,375,43,52,125,51,97
105 DATA25Ø,57,172,25Ø
110 DATA51,97,5ØØ,Ø,Ø,125,43,52,25Ø,51,97,25Ø,57,172
115 DATA1ØØØ,51,97,5ØØ
```

CREANDO EFECTOS ESPECIALES

Contrariamente a la música, los efectos especiales tiene otros objetivos, como por ejemplo la explosión de una nave espacial en un juego de marcianos...o una sirena de alarma en un programa de gestión, cuando Ud. está a punto de borrar accidentalmente un disco.

Tiene una amplia gama de posibilidades (disponibles) si quiere crear diversos efectos sonoros especiales. Aqui hay 10 ideas programables para que se pueda familiarizar con los efectos sonoros:

1. Cambie el volumen, cuando se toca una nota, por ejemplo para crear un efecto de eco.
2. Varíe entre dos notas rápidamente para crear un efecto de "trémolo".
3. Ajuste la modulación en diversas voces.
4. El Ataque/Caída...para alterar la relación en sonido sube a su nivel pico y vuelve a caer.
5. El SOSTENIMIENTO/RELAJACION, para alargar o disminuir la duración de una nota, o para cambiar diversos tipos de sonidos pruebe varios ajustes.
6. Efectos de VOCES MULTIPLES...tocando más de una voz al mismo tiempo, y controlada independientemente y tocando más o menos al tiempo que otra, o haciendo un eco a la primera, obtendrá efectos fantásticos...
7. Cambie notas de la escala musical o de la octava utilizando la tabla de notas musicales.
8. Utilice la modulación cuadrada o ajustes de pulso para diversos efectos.
9. Utilice la modulación de ruido para crear un RUIDO blanco, para acentuar ciertos sonidos, crear explosiones, tiros de metralleta, pasos... Las mismas notas musicales sirven para crear diferentes tipos de Ruidos Blancos.
10. Combine diversas frecuencias Altas y Bajas en sucesión rápida, en diferentes octavas.
11. El Filtro, pruebe este otro control con el ajuste de POKE, que viene en el Apéndice M.



... DE EFECTOS SONOROS ESPECIALES

Estos programas que vamos a ver pueden añadirse a **todo** programa BASIC. Se los ofrecemos para darle una idea de programar y mostrarle las posibilidades del Commodore 64 en materia de efectos especiales.

Observe las abreviaciones que hemos hecho al principio del programa en la línea 10. En efecto, se pueden abreviar los poco significativos y prácticos números de control, entrándolos por medio de variables, con letras más significativas. La línea 10 sólo sirve para que a esas letras sencillas de entender, que son las variables, se les asigne los largos números de código de los controles. Aquí V es Volumen, W = Forma de onda, AC = Ataque/Decaimiento, H = Alta Frecuencia (VOZ 1) y L = Baja Frecuencia (VOZ 1). Así pués utilizaremos estas letras en vez de utilizar los números correspondientes en nuestro programa, haciéndolo así más corto, más manejable, más fácil de entrar y cambiar, si se deseara.

GRITO DE MUÑECAS:

```
10 V=54296:W=54276:A=54277:H=54273:L=54272
20 POKEV,15:POKEW,65:POKEA,15
30 FORX=200TO5STEP-2:POKEH,40:POKEL,X:NEXT
40 FORX=150TO5STEP-2:POKEH,40:POKEL,X:NEXT
50 POKEW,0
```

SONIDO DE ESTALLIDO UTILIZANDO LA VOZ 1, LA MODULACION DE RUIDO, Y CAIDA DEL VOLUMEN.

```
10 V=54296:W=54276:A=54277:H=54273:L=54272
20 FORX=15TO0STEP-1:POKEV,X:POKEW,129:POKEA,15:
   POKEH,40:POKEL,200:NEXT
30 POKEW,0:POKEA,0
```

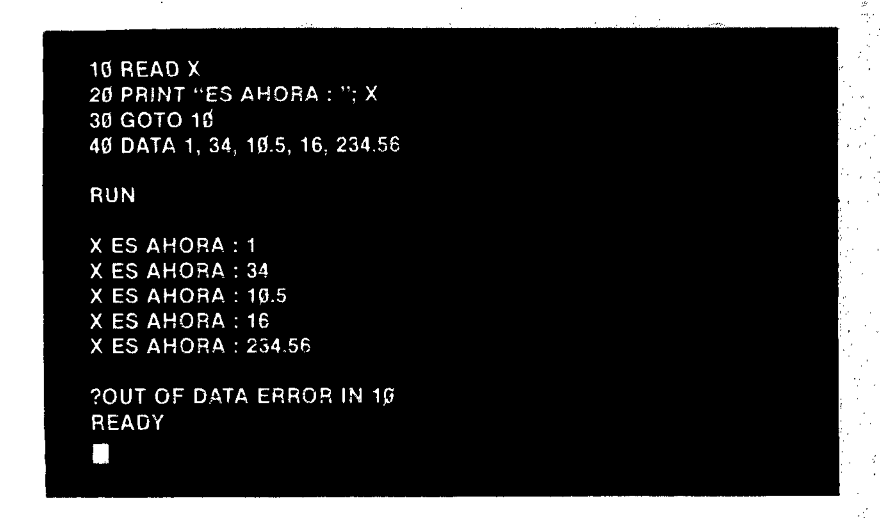

EL MANEJO AVANZADO DE DATOS

LAS INSTRUCCIONES READ Y DATA

Ya ha visto como asignar valores a las variables, directamente en un programa (A = 2), y como asignarle a una variable diversos valores durante la ejecución del programa-vía la instrucción INPUT.

Pero hay muchas ocasiones, demasiadas, en las que ningúno de estos procedimientos son adecuados al trabajo que queremos realizar, especialmente si hay mucha información que manejar.

Pruebe este programa corto:



```
10 READ X
20 PRINT "ES AHORA : "; X
30 GOTO 10
40 DATA 1, 34, 10.5, 16, 234.56

RUN

X ES AHORA : 1
X ES AHORA : 34
X ES AHORA : 10.5
X ES AHORA : 16
X ES AHORA : 234.56

?OUT OF DATA ERROR IN 10
READY
■
```

En la línea 10, el ordenador lee (READ) un valor de la instrucción DATA, y asigna este valor a X. Cada vez que se pasa por el bucle el valor siguiente del "DATA" es leído y su valor asignado a X, y luego escrito (por PRINT). Un puntero en el mismo ordenador, lleva la cuenta del valor que ha de ser leído después.

PUNTERO
↓
4Ø DATA 1, 34, 1Ø.5, 16, 234.56

Cuando se han utilizado todos los valores, y el ordenador ejecuta el bucle de nuevo, buscando otros datos, aparece el mensaje de error OUT OF DATA indicando que no hay más valores que leer.

Es importante seguir el formato de la instrucción DATA de manera precisa.

4Ø DATA 1, 34, 1Ø.5, 16, 234.56

↑ Coma para separar cada valor — ↑ Sin coma (o punto)

La instrucción DATA puede contener, números enteros, reales (235,56), o números expresados con la notación científica. Pero no puede leer otras variables, o operaciones aritméticas en las líneas de DATA. Lo siguiente sería incorrecto:

4Ø DATA A, 25/36, 2*5

Pero lo que sí puede hacer es utilizar una variable alfanumérica en una instrucción READ y luego poner cadenas de texto o alfanuméricas en las líneas de DATA. Lo siguiente es perfectamente aceptable:

```
NEW

10 FOR X = 1 to 3
15 READ A$
20 PRINT "A$ ES AHORA : "; A$
30 NEXT
40 DATA ESTO, ES, DIVERTIDO

RUN

A$ ES AHORA : ESTO
A$ ES AHORA : ES
A$ ES AHORA : DIVERTIDO
READY
```

Note que ahora, la instrucción READ está situada dentro de un BUCLE FOR ...NEXT. Este bucle se ejecuta para indicar el número de valores que tiene el "DATA".

En muchos casos, cambiará el número de valores de la instrucción DATA, tantas veces como ejecute el programa. Una manera de evitar el tener que contar el número de valores de las líneas de DATA e incluso de que se presente el mensage de error OUT OF DATA es, situar un FLAG o BANDERA como último valor de una línea de DATA. Este podría ser un valor que el data nunca igualaría, como por ejemplo un número negativo o un número muy pequeño o grande. Cuando este valor sea leído, la ejecución del programa será transferida a la parte siguiente.

Hay una manera de reutilizar los mismos DATA otra vez en el programa uti-

lizando la introducción RESTORE que "restaura" (resitua) el puntero de DATA al principio de la lista de DATOS. Añada la línea 50 al programa anterior:

5Ø GOTO 1Ø

Volverá a obtener un "OUT OF DATA ERROR", puesto que como el programa se va a la línea 10 para RELEER LOS DATOS, el puntero indicará que todos los datos han sido utilizados. Ahora, añada:

45 RESTORE

y lance el programa de nuevo. El "puntero" de DATA ha sido RESTAURADO y los DATOS pueden ser leídos continuamente.

LAS MEDIAS ARITMETICAS

El programa siguiente ilustra una utilización práctica de los READ y DATAS, leyendo una serie de números y luego calculan su media.

```
NEW

5 T=Ø:CT=Ø
1Ø READX
2Ø IFX=-1THEN5Ø:REM PRUEBA PARA FIN
25 CT=CT+1
3Ø T=T+X:REM DATOS SOBRE TOTAL
4Ø GOTO1Ø
5Ø PRINT"SON ";CT;"VALORES LEIDOS"
6Ø PRINT"TOTAL = ";T
7Ø PRINT"MEDIA = ";T/CT
8Ø DATA75,8Ø,62,91,87,93,78,-1

RUN
SON 7 VALORES LEIDOS
TOTAL = 566
MEDIA = 8Ø.8571429
```

La línea 5 pone a cero, el Contador CT y el T, de Total. La línea 10 lee (READX) un valor y lo asigna a X. La línea 20 mira si este valor es el del flag, (o bandera, aquí-1). Si el valor leído forma parte de los valores válidos de la instrucción DATA, CT se incrementa en 1 y se añade X al total.

Cuando la bandera es leída, el programa va a la línea 50 que escribe en pantalla el número de valores que han sido leídos.



La línea 60 escribe el total, y la línea 70 la divide por el número de valores para obtener así, la media aritmética.

Utilizando una bandera al final de una línea de DATA puede Ud. poner cualquier número de valores en la instrucción de DATA, que puede superar varias líneas —sin tener que preocuparse del número de valores que se han entrado dentro.

Otra variante de la instrucción READ concierne el asignar información de una misma línea de DATA a diferentes variables. Esta información puede incluso ser una mezcla de valores numéricos y de cadenas de texto. Puede hacer todo esto con el siguiente programa, que leerá un nombre, algunas puntuaciones —por ejemplo de bolos— y luego escribirá su nombre, puntuación y media:

```
NEW

1Ø READN$,A,B,C
2Ø PRINTN$;" TIENE UNA PUNTUACION DE: ";A;" ";B;" ";C
3Ø PRINT"Y SU MEDIA ES: ";(A+B+C)/3
4Ø PRINT:GOTO1Ø
5Ø DATAMIGUEL,19Ø,185,165,JOSE,225,245,19Ø
6Ø DATAJUAN,155,185,2Ø5,PABLO,16Ø,179,187

RUN

MIGUEL TIENE UNA PUNTUACION DE: 19Ø  185  165
Y SU MEDIA ES: 18Ø

JOSE TIENE UNA PUNTUACION DE: 225  245  19Ø
Y SU MEDIA ES: 22Ø
```

Al efectuar el programa, las instrucciones DATA estaban dispuestas en el mismo orden en que la instrucción READ esperaba la información: un nombre (una cadena), y los 3 valores. En otras palabras, la primera vez N$ saca el dato MIGUEL. READ A correspondiente a 190 en la instrucción DATA, B a 150 y C a 165. El proceso se repite en este orden para la información restante. (JOSE y su puntuación, JUAN y la suya, y PABLO con la suya).

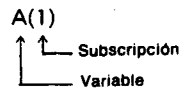
LAS VARIABLES SUSCRITAS

En el pasado hemos utilizado sencillamente las clásicas variables del BASIC, como A, A$ y NU para representar valores. Estas eran sencillas letras se-



guidas por otra letra o una sola cifra. En muchos de los programas que escribirá, es casi imposible que necesitemos más variables de las que se pueden obtener con la combinación de dos números o letras posibles. Pero está Ud. limitado, en el sentido en que se utilizan las variables en un programa.

Así pués, dejemos introducir el concepto de la variable suscrita:



Esto se diria: A sub 1. Una variable subscrita consiste en una letra seguida de un número encerrado entre paréntesis. Por favor dese cuenta de la diferencia que hay entre A, A1 y A(1). Todas son variables únicas, pero sólo A(1) es una variable subscrita.

Las variables subscritas, de la misma manera que las variables corrientes, bautizan una situación de memoria del ordenador. Piense en una variable subscrita como en casillas para guardar información, como con las variables normales.

A(0)	
A(1)	
A(2)	
A(3)	
A(4)	

Si escribió:

```
10 A(0) = 25: A(3) = 55 : A(4) = —45.3
```

entonces la memoria tendrá este aspecto.

A(0)	25
A(1)	
A(2)	
A(3)	55
A(4)	−45.3

Este grupo de variables subscritas también se puede llamar una matriz. En este caso una matriz unidimensional. Más tarde introduciremos matrices multidimensionales.

Las variables subscritas pueden ser complejas incluyendo otras variables o cálculos. Las variables siguientes son válidas:

A(X) A(X + 1) A(2 + 1) A(1*3)

Las expresiones con paréntesis se calculan según las mismas reglas de operaciones aritméticas que en el Capítulo 2.



Ahora que las reglas básicas estan asentadas, ¿como se pueden utilizar las variables subscritas? Una manera es almacenar una lista de números entrados con INPUT o READ.

Utilicemos las variables subscritas para calcular medias de otra forma.

```
5 PRINTCHR$(147)
1Ø INPUT"CUANTOS NUMEROS :";X
2Ø FORA = 1TOX
3Ø PRINT"ENTRE VALOR # ";A;:INPUTB(A)
4Ø NEXT
5Ø SU = Ø
6Ø FORA = 1TOX
7Ø SU = SU + B(A)
8Ø NEXT
9Ø PRINT:PRINT"MEIDA = ";SU/X


RUN

CUANTOS NUMEROS :? 5
ENTRE VALOR # 1  ? 125
ENTRE VALOR # 2  ? 167
ENTRE VALOR # 3  ? 189
ENTRE VALOR # 4  ? 167
ENTRE VALOR # 5  ? 158

AVERAGE =  161.2
```

Puede que haya una manera más sencilla de realizar lo que hicimos en este programa, pero este ilustra como trabajan las variables subscritas. La línea 10 pide "cuantos números" van a ser entrados. Esta variable, X, actúa como controlador del bucle, con los valores entrados y asignados a la variable subscrita B.

Cada vez que se pasa por el bucle de INPUT, A se incrementa de 1 y así el próximo valor que se entra será asignado al elemento siguiente de la matriz A. Por ejemplo, la primera vez que pasa por el bucle A = 1, el primer valor es asignado a B(1). La vez siguiente A = 2 y el valor siguiente se asigna a B(2) y así hasta que todos los valores hayan sido entrados.

Pero ahora viene una gran diferencia. Una vez que se han entrado todos los valores, se almacenan en una matriz, lista para trabajar de varias maneras. Antes, llevaba la cuenta del total, cada vez que se pasaba por el bucle de IN-



PUT o READ, pero nunca se podrá volver a los datos iniciales sin volver a leer de nuevo la información.

De las líneas 50 a 80 se encuentra, otro bucle, diseñado para sumar los diversos elementos de la matriz y luego presentar la media. Esta parte separada del programa muestra que todos los valores se almacenan y se puede disponer de ellos cuando se necesite.

Para comprobar que todos los valores individuales están realmente almacenados por separado en la matriz, mecanografíe lo siguiente inmediatamente después de haber lanzado el programa anterior.

FOR A = 1 TO 5 : ?B(A),: NEXT

```
125      1Ø7      189      167
158
```

La pantalla le presentará sus valores actuales de la manera en que el contenido de la matriz ha sido escrito.

LAS DIMENSIONES

Si trata de entrar más de 10 números en el ejemplo anterior le saldrá un ERROR de DIMENSION. Matrices de más de 11 elementos (subscritos de 0 a 10) para una matriz unidimensional deben utilizarse, donde sea necesario, de la misma manera en que las variables nulas pueden utilizarse en cualquier lugar de un programa. Matrices de más de 11 elementos deben ser declaradas en una instrucción de dimensión.

Añada esta línea al programa:

5 DIM B(1ØØ).

Esto indica al ordenador que tendrá un máximo de 100 elementos en la matriz.

La instrucción de dimensión puede utilizarse también con una variable, así la línea siguiente podría reemplazar la línea 5 (no se olvide de eliminar la línea 5):

15 DIM B(X)

Esto dimensionará la matriz con el número exacto de valores que se entrarán.

¡Tenga cuidado! Una vez formateada, una matriz no puede ser redimensionada en otro lugar del programa. Pero puede, sin embargo, tener varias matrices en el programa y dimensionarlas todas en la misma línea, así:

1Ø DIM C(2Ø), D(5Ø), E(4Ø)

SIMULACION DEL LANZAMIENTO DE DADOS, MATRICES

A medida que los programas se vuelven más complejos, utilizando las variables subscritas, abreviará el número de instrucciones necesarias, y simplificará la escritura del programa.

Un sencillo variable subscrita puede ser utilizada, por ejemplo, para llevar la cuenta del número de veces que sale una misma cara.

```
1 REM SIMULACION DADO
10 INPUT"CUANTAS TIRADAS ";X
20 FORL=1TOX
30 R=INT(6*RND(1))+1
40 F(R)=F(R)+1
50 NEXTL
60 PRINT"CARA","NUMERO DE TIRADAS"
70 FORC=1TO6:PRINTC,F(C):NEXT
```

La matriz F se utilizará para llevar la cuenta del número de veces que sale una misma CARA del dado. Por ejemplo cada vez que sale un 2 F(2) se incrementa en 1. Utilizando el mismo elemento de la matriz para llevar el número actual de la cara qué ha salido, hemos eliminado la necesidad de 5 variables más (una para cada cara) y numerosas instrucciones para ver y revisar que número se ha sacado.

La línea 10 pide cuantas tiradas quiere simular.

La línea 20 establece el bucle para crear el número aleatorio del dado e incrementa el elemento apropiado de la matriz en 1 para cada tirada.

Una vez que se hayan lanzado los dados el número de veces requerido, la línea 60 escribe los títulos y la línea 76 escribe el número de veces que ha salido cada cara.

Un ejemplo de ello, tendría este aspecto:

```
CUANTAS TIRADAS: ? 1ØØØ
CARA                NUMERO DE TIRADAS
1                   148
2                   176
3                   178
4                   166
5                   163
6                   169
```

¡Bien, pués al fin y al cabo no estaba cargado!

A título de comparación, lo siguiente es otra manera de re-escribir este programa, pero sin utilizar las variables subscritas. No se preocupe en entrarlo, pero observe el número de instrucciones adicionales que han sido necesarias.



```
1Ø INPUT "CUANTAS TIRADAS ";X
2Ø FOR L = 1 TO X
3Ø R = INT(6*RND(1))+1
4Ø IF R = 1 THEN F1 = F1 + 1 : NEXT
41 IF R = 2 THEN F2 = F2 + 1 : NEXT
42 IF R = 3 THEN F3 = F3 + 1 : NEXT
43 IF R = 4 THEN F4 = F4 + 1 : NEXT
44 IF R = 5 THEN F5 = F5 + 1 : NEXT
45 IF R = 6 THEN F6 = F6 + 1 : NEXT
6Ø PRINT "CARA", "NUMERO DE TIRADAS"
7Ø PRINT 1, F1
71 PRINT 2, F2
72 PRINT 3, F3
73 PRINT 4, F4
74 PRINT 5, F5
75 PRINT 6, F6
```

El programa se ha doblado en volumen, de 8 a 16 líneas. En programas largos el espacio ahorrado con las variables subscritas podría ser aún mayor.

MATRICES BI-DIMENSIONALES

Antes en este mismo capítulo hemos experimentado con matrices de una sola dimensión. Este tipo de matriz se representaba como un grupo de casillas consecutivas con memoria que contenían un elemento de la matriz. ¿Como espera que se represente una matriz de dimensión 2?.

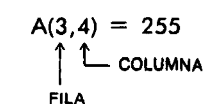
Primero una matriz bi-dimensional se escribirá así:



y se puede representar como una malla:

	Ø	1	2	3	4	5	6
Ø							
1							
2							
3							
4							

La suscripción puede representarse como una fila y columna en la tabla donde se encuentra almacenado el elemento particular de la matriz.



	Ø	1	2	3	4	5	6
Ø							
1							
2							
3					255		
4							

Si asignamos el valor 255 a A(3,4), se puede concebir como que ha sido situado en la 3.ª fila y 4.ª columna de la tabla.

Las matrices bi-dimensionales se comportan de acuerdo que las mismas reglas establecidas con las matrices uni-dimensionales.

Deben ser dimensionadas	DIM A(2Ø,2Ø)
Asignación de datos	A(1,1) = 255
Asignar valores a otras variables	AB = A(1,1)
Escribe valores	PRINT A(1,1)

Si las matrices de 2 dimensiones trabajan como sus compañeras de una dimensión, ¿que ventajas adicionales tiene el crear matrices multi-dimensionales?

Pruebe esto: ¿Ve alguna manera de utilizar una matriz bi-dimensional para tabular los resultados de un cuestionario para su club en que intervengan 4 preguntas, y en que puede haber hasta 3 respuestas por cada una de ellas? El problema puede representarse así:

CUESTIONARIO DEL CLUB:

PR1: ESTA UD. A FAVOR DE LA MOCION NO. 1?

☐ 1-SI ☐ 2-NO ☐ 3-INDECISO

...y así.



La tabla de la matriz de este problema puede representarse así:

	RESPUESTAS		
	SI	NO	INDECISOS
PREGUNTA 1			
PREGUNTA 2			
PREGUNTA 3			
PREGUNTA 4			

El programa para establecer la tabulación actual, para el cuestionario debe tener la forma de lo que se muestra en la página 103.

Este programa contiene todas las técnicas de programación que hemos visto hasta aquí. Incluso si no tiene ninguna necesidad de ver el programa actual, en estos momentos, mire si puede ver como funciona.

El corazón de este programa es una matriz bidimensional de 4x3 A(4,3). El número total de respuestas para cada respuesta posible de cada pregunta está contenida en el apropiado elemento de la matriz. Para simplificar el asunto, no utilizamos las primeras líneas y columnas (A(0,0) a A(0,4)). Recuerde que estos elementos siempre están presentes en cualquier matriz que diseñe.

En la práctica, si se responde "si" a la pregunta I, entonces A(1,1) es incrementado de 1 —la fila 1 por la pregunta 1, y la columna 1 para la respuesta "si". El resto de las preguntas y respuestas siguen el mismo patrón. Una respuesta negativa (NO) para la pregunta 3 añadirá uno al elemento A(3,2) y así...

SHIFT

```
2Ø PRINT"(CLR/HOME)"
3Ø FOR R = 1 TO 4
4Ø PRINT"PREGUNTA # : "; R
5Ø PRINT " 1—SI  2—NO  3—INDECISO"
6Ø PRINT "CUAL ES SU RESPUESTA : ";
61 GET C : IF C <1 or C>3 THEN 61
65 PRINT C: PRINT
7Ø A(R,C) = A(R,C) + 1: REM DATOS SOBRE EL ELEMENTO
8Ø NEXT R
85 PRINT
9Ø PRINT "QUIERE ENTRAR OTRO": PRINT "RESPONDA (S/N)";
1ØØ GET A$ : IF A$ = "" THEN 100
11Ø IF A$ = "S" THEN 2Ø
12Ø IF A$ <> "N" THEN 1ØØ
13Ø PRINT "(CLR/HOME)";"TODAS LAS RESPUESTAS SON:":PRINT
14Ø PRINT  SPC(18);"RESPUESTA"
141 PRINT "PREGUNTA","SI","NO","INDECISO"
142 PRINT"..............................................................................."
15Ø FOR R = 1 TO 4
16Ø PRINT R, A(R,1), A(R,2), A(R,3)
17Ø NEXT R
RUN

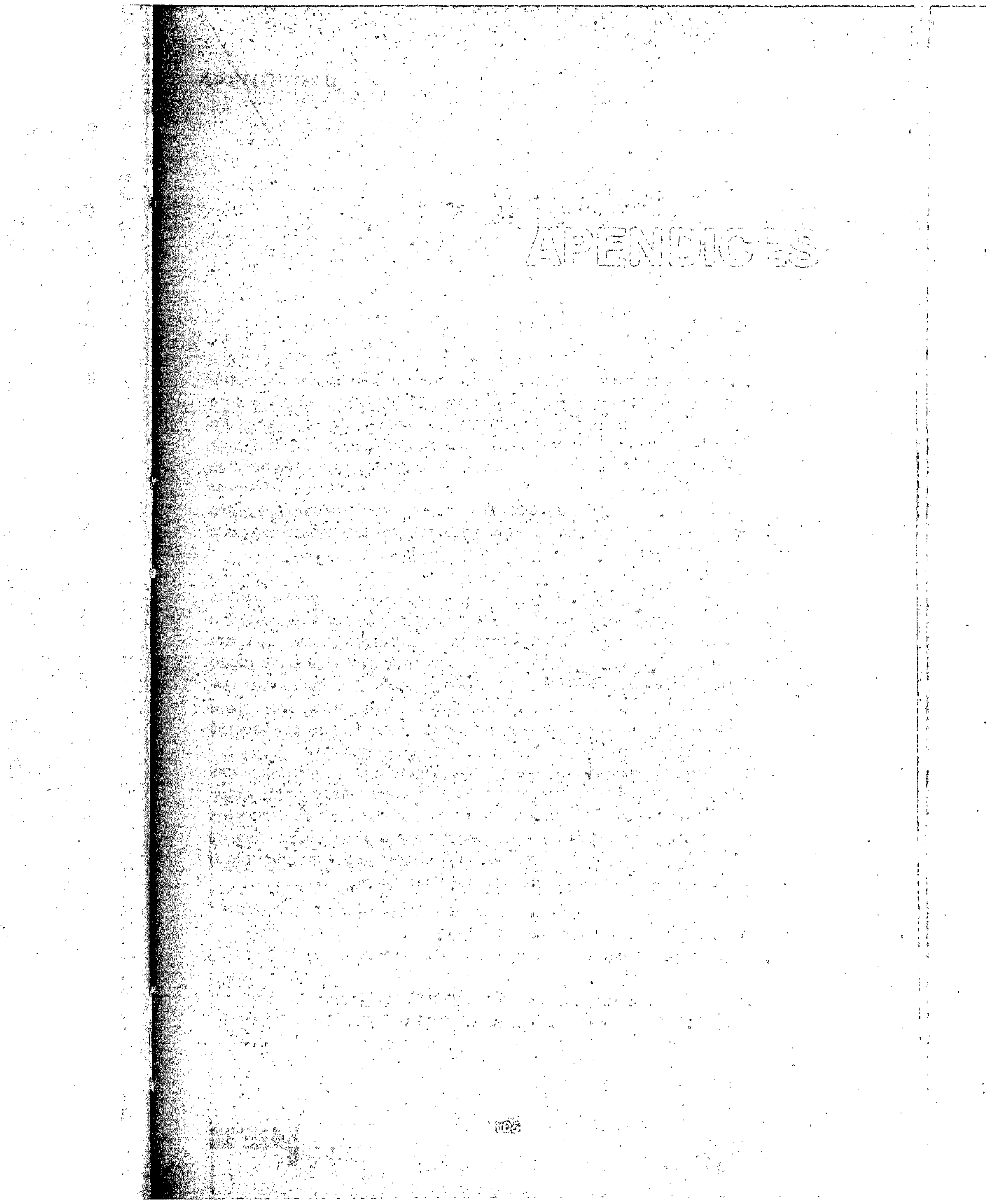
PREGUNTA # : 1
1—SI  2—NO  3—INDECISOS
CUAL ES SU RESPUESTA : 1

PREGUNTA # : 1
CUAL ES SU RESPUESTA : 1

Y por tanto...

TODAS LAS RESPUESTAS SON:
```

		RESPUESTA	
PREGUNTA	SI	NO	INDECISOS
1	6	1	Ø
2	5	2	Ø
3	7	Ø	Ø
4	2	4	1



APENDICES

INTRODUCCIÓN

Ahora que se ha familiarizado íntimamente con su Commodore 64, queremos que sepa que nuestra ayuda al cliente no acaba aquí. Es probable que no lo sepa, pero Commodore esta en el mercado desde hace más de 23 años. En los años 70 sacamos el primer ordenador personal autónomo, el PET. Desde entonces nos hemos vuelto la compañía de ordenadores n.º 1 en muchos países del mundo (como en España). Nuestra capacidad de diseñar y fabricar nuestros propios "chips" del ordenador nos permite brindarle mejores y más avanzados ordenadores personales, a unos precios muy por debajo de lo que podía esperar para equipos de este alto nivel técnico.

Commodore se propone no solamente apoyarle a Ud. el usuario, sino también al vendedor al que le compró este equipo, con revistas que publiquen artículos de divulgación técnica, de aplicaciones... y lo más importante, apoyar a la gente que desarrolle software, los programas en cartucho, disco o cassette para utilizarlos con su ordenador. Le aconsejamos que se ponga en contacto con algún club de usuarios de Commodore, donde podrá aprender nuevas técnicas, intercambiar ideas y hacer descubrimientos. Publicamos una revista que contiene consejos de programación, información sobre nuevos productos e ideas para aplicaciones con su ordenador. (Vea el apéndice N).

Los apéndices siguientes contienen tablas, listas y otras informaciones que le permitirán programar el Commodore 64 de manera más eficiente y rápida. También viene incluida información sobre la gran variedad de productos Commodore en los que puede estar interesado, y una bibliografía con más de 20 libros y revistas que le ayudarán a desarollar programas y le mantendrán al corriente sobre las novedades en materia de ordenadores.

APENDICE A:

Los ACCESORIOS del COMMODORE 64 y el SOFTWARE

LOS ACCESORIOS

El Commodore 64 puede utilizar las unidades de almacenamiento y otros accesorios del VIC-20 de Commodore —la unidad de cassette: Datassette, la unidad de disco, la impresora— de manera que su sistema puede ampliarse según sus necesidades...

- LA UNIDAD DE CASSETTE: Esta unidad de bajo coste, le permite almacenar en una cassette programas y datos para utilizarlos posteriormente. Esta unidad también puede servir para leer programas pre-grabados.
- LA UNIDAD DE DISCOS: Esta unidad de disco único utiliza discos o floppy diskettes standard de 5 ¼ pulgadas, que se asemejan a un disco de 45 RPM, para almacenar programas y datos. Los discos le permiten un acceso a la información mucho más rápido y pueden llevar hasta 170.000 caracteres cada uno. Las unidades de disco son inteligentes es decir que tienen su propio microprocesador y memoria. Los discos no necesitan ninguna memoria del Commodore 64.
- LA IMPRESORA: La impresora VIC le ofrece copias de programas, datos y gráficos. Esta impresora de matriz de puntos de 30 caracteres/segundo, utiliza papel corriente perforado y otros suministros de bajo coste. La impresora se conecta directamente al Commodore 64 sin ningún interface adicional.
- CARTUCHOS DE INTERFACE: Un cierto número de cartuchos especializados estarán disponibles para el Commodore 64 para permitirle conectar con varias unidades estandard como impresoras, controladores, e instrumentación que se puedan acoplar al sistema.

  Con un cartucho especial IEEE-488, el Commodore 64 podrá utilizar toda la gama de periféricos CBM incluyendo unidades de discos e impresoras.

  Además, un cartucho Z80 le permitirá utilizar programas en CP/M* en el Commodore 64, abriéndole las puertas a la mayor base de aplicaciones posibles.

[illegible]

Le ofrecemos varias categorías de software para su Commodore 64, una profesional, otra de juegos, y otra didáctica:

AYUDAS PROFESIONALES:

- El paquete electrónico de "spreadsheet" le permitirá planear presupuestos y practicar el análisis del tipo "Que pasaría si". Y con un programa gráfico opcional, se podrán crear gráficas significativas a partir de los datos del "spreadsheet".
- La Planificación Financiera, como la amortización de préstamo, podrán ser llevadas a cabo con el Paquete "Planificación Financiera".
- Un cierto número de programas de dirección profesional de tiempos ayudará a la dirección.
- Los programas de BASE de DATOS, de manejo sencillo, le permitirán llevar con mucha información fácilmente listas de correos, listines telefónicos, inventarios o/y cualquier tipo de información organizada de manera práctica.
- Los programas de "procesos de texto" convertirán su Commodore 64 en un procesador de texto con todas sus posibilidades. El redactar, corregir o adaptar memorias, cartas, u otros textos no será más que un juego.

DIVERSION, ENTRETENIMIENTO Y JUEGOS

- Los juegos de mejor calidad estarán disponibles en cartuchos para el Commodore 64, ofreciéndole horas y horas de diversión. Estos programas utilizan la alta resolución gráfica, y todas las posibilidades de efectos sonoros del Commodore 64.

---

*CP/M es una marca registrada de Digital Research Inc.

DIDACTICOS / EDUCATIVOS

El Commodore 64 es un tutor que no se cansa y que le ofrece siempre sus atenciones personales. Además de la gran variedad de programas educativos para el PET, estarán disponibles además, otros lenguajes educativos para el Commodore 64 como el "PILOT" y el "LOGO" y otros paquetes avanzados claves.

APENDICE B:

MANEJO AVANZADO de la UNIDAD del CASSETTE

Además de guardar copias de sus programas en cintas, el Commodore 64 puede también almacenar los valores de variables y otro tipo de datos, en un grupo llamado FICHERO. Esto le permite almacenar incluso mucha más información de la que cabe en la memoria del ordenador al mismo tiempo.

Las instrucciones que se utilizan con los FICHEROS de DATOS son: OPEN, CLOSE, PRINT # y GET #. El sistema de variable ST (estado) sirve para revisar las marcas de la cinta.

Cuando se escribe sobre cinta se emplean los mismos conceptos que cuando se escribe en pantalla. Pero, en vez de escribir la información en la pantalla, se graba en la cinta utilizando una variante de la instrucción PRINT —el PRINT #.

El programa siguiente ilustra como funciona esto:

```
10 PRINT"PROGRAMA DE ESCRITURA EN CINTA"
20 OPEN1,1,1,"FICHERO DATOS"
30 PRINT"ESCRIBA DATOS PARA ALMACENAR O ESCRIBA STOP"
50 PRINT
60 INPUT"DATO";A$
70 PRINT#1,A$
80 IFA$<>"STOP"THEN50
90 PRINT
100 PRINT"CERRANDO FICHEROS"
110 CLOSE1
```

La primera cosa que debe hacer es ABRIR (OPEN) el fichero (en este caso FICHERO DE DATOS). De esto se ocupa la línea 1Ø.

El programa le pregunta por los datos que quiere grabar sobre cinta en la línea 6Ø. La línea 7Ø escribe lo que mecanografía —dentro de A$— en la cassette. Y el proceso continúa.

Si escribe STOP, la línea 11Ø CIERRA el fichero.

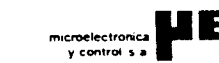
Para leer, la información rebobine la cinta y pruebe esto:

```
10 PRINT"PROGRAMA LEER EN CINTA"
20 OPEN1,1,0,"FICHERO DATOS"
30 PRINT"FICHERO ABIERTO"
40 PRINT
50 INPUT#1,A$
60 PRINTA$
70 IFA$="STOP"THENEND
80 GOTO40
```

De nuevo el fichero "FICHERO DE DATOS" debe ABRIRSE (OPEN). En la línea 5Ø el programa entra (INPUT) A$ de la cuenta y lo imprime (PRINT) en la pantalla. Luego el proceso se repite de nuevo hasta encontrar un "STOP" lo que para el programa.

Una variante de GET, GET #, también puede utilizarse para leer los datos de la cinta. Remplace las líneas 5Ø a 8Ø por las siguientes:

```
50 GET#1,A$
60 IFA$=""THENEND
70 PRINTA$,ASC(A$)
80 GOTO50
```

APENDICE C:

BASIC del COMMODORE 64

En la primera parte de este manual se ha dado una introducción al lenguaje BASIC usado por el C 64 suficiente para que usted tenga una idea de los medios usados para la programación de ordenadores y haya adquirido ya cierta práctica en su utilización. En los apéndices incluidos a continuación se amplía esta información, especialmente en el presente Apéndice, en que se relacionan las reglas (SINTAXIS) del lenguaje BASIC empleado en su ordenador C 64, con una breve descripción de cada elemento del lenguaje. Usted debería ensayar el uso de estos comandos e instrucciones, ya que aun cuando se equivoque no puede causar daños a su ordenador, y la mejor forma de aprender el uso de un ordenador y su programación es practicando ambos.

Este Apéndice está dividido en secciones, para relacionar los diferentes tipos de operaciones BASIC:

1. **Variables y operaciones:** Esta sección describe los diferentes tipos de variables, nombres legales de variables y operadores aritméticos y lógicos.
2. **Comandos:** Esta sección describe los comandos utilizados en programas y su edición, almacenamiento y borrado.
3. **Instrucciones:** *Esta sección describe las instrucciones disponibles para* componer programas en BASIC. Las instrucciones se diferencian de los comandos en que se les antepone normalmente un número de líneas.
4. **Funciones:** Esta sección describe las funciones, divididas en tres categorías: numéricas, tipo cadena y funciones usadas con la instrucción PRINT.

Los comandos están relacionados en cada sección por orden alfabético, para facilitar su consulta. Una exposición más detallada de estos términos se encuentra en la Guía de Referencia del Programador, suministrada por el proveedor del equipo Commodore 64.

VARIABLES Y OPERADORES

a. VARIABLES

El Commodore 64 usa tres tipos de variables en BASIC: *variables numéricas normales, numéricas de enteros y de tipo cadena (alfanuméricas)*

Las variables numéricas y normales, se denominan también variables de coma decimal flotante y pueden tener cualquier valor desde -10^{38} a $+10^{38}$, con una precisión de nueve dígitos. Cuando un número sea mayor de nueve dígitos, como 10^{10} ó 10^{-10}, el ordenador lo visualizará en notación científica, con el número normalizado a 1 dígito y ocho decimales, seguidos de la letra E (exponenciación) y la potencia de diez por la que debe multiplicarse el número: por ejemplo; el número 12345678901 se representará en la pantalla como 1.23456789E + 11.

Las variables numéricas de enteros, se usan cuando el número esté siempre comprendido entre +32767 y -32768 y carezca de fracciones. Las variables de enteros requieren menos memoria que las variables de coma decimal flotante y normalmente la diferencia es de poca consideración, a menos que se trate de una gran cantidad, como en "tablas" (véase pág. 113). Las variables *de enteros son números tales como* 3, 9 ó -100.



Las variables tipo cadena, son usadas para datos y pueden contener números, letras y otros caracteres posibles con el C 64. Un ejemplo de variables de cadena es "Commodore 64".

Los nombres de variables pueden estar integrados por una sola letra, una letra seguida de un número o dos letras.

Las variables de enteros se especifican usando el signo de porcentaje (%) después del nombre de la variable. *Las variables de cadena se distinguen con* el sufijo dólar ($).

EJEMPLOS: Nombres de variables numéricas: A, A5, BZ
Nombres de variables de enteros: A%, A5%, BZ%
Nombres de variables de cadena: A$, A5$, BZ$

Las *"tablas"* son listas de variables con un mismo nombre, cada una seguida de un número para especificarla dentro del conjunto. Las tablas son definidas con la instrucción DIM y pueden contener variables de coma decimal flotante, de enteros o tipo cadena. El nombre de la variable en la tabla va seguida por paréntesis, en el que se coloca el número de la variable en la lista.

EJEMPLOS: A(7),BZ%(11),A$(87).

Las tablas pueden tener más de una dimensión. Una tabla bidimensional puede considerarse que tiene filas y columnas; el primer número entre paréntesis indica la fila y el segundo la columna.

EJEMPLOS: A(7,2),BZ%(2,3,4),Z$(3,2)

Hay tres nombres de variables que son reservados para uso del C 64 y no pueden utilizarse en el trabajo normal; son las variables ST, TI y TI$. ST es una variable indicadora del estado de operaciones de entrada y salida; el valor de ST cambia si hay anomalías en la carga de un programa o datos desde cinta cassette o disco floppy. Las diversas formas de esta función se explican más detalladamente en el Manual de Referencia de Programación.

Las variables reservadas TI y TI$ se usan con el reloj de tiempo real incorporado en el C 64. La primera es actualizada cada 1/60 de segundo. Empieza en cero cuando se conecta el C 64 y puede volverse a cero sólo cambiando el valor de TI$. Esta última es una cadena que es actualizada constantemente por el sistema. Los dos primeros caracteres constituyen el número de horas, los caracteres tercero y cuarto el número de minutos y los caracteres quinto y sexto el número de segundos. A esta variable se le puede adjudicar cualquier valor (a condición de que todos los caracteres sean números) y se mantendrá actualizada a partir de este punto.

EJEMPLO:TI$ = "101530" ajustable el reloj a 10:15 y 30 seg. (de la mañana)

Este reloj se borra cuando se desconecta el C 64 y vuelve a empezar a cero cuando se conecta.

b. OPERADORES

Los operadores aritméticos incluyen los signos siguientes:

\+ suma
— resta
* multiplicación
/ división
↑ elevación a potencia (exponenciación)



Cuando en una línea hay más de un operador, todos ellos se ejecutan en el siguiente orden de prioridad: primero, exponenciación; a continuación, multiplicación y división, y, finalmente, suma y resta. Si usted quiere que las operaciones se ejecuten en otro orden del preprogramado en la máquina, debe incluir entre paréntesis aquellas operaciones que deban realizarse antes; sin embargo, es preciso que en tales fórmulas haya el mismo número de paréntesis de apertura que de cierre, de lo contrario en la pantalla aparecerá un mensaje de error: SYNTAX ERROR.

Hay también operadores para igualdad o desigualdad:

=	igual a
<	menor que
>	mayor que
< = ó = <	menor que o igual a
> = ó = >	mayor que o igual a
< > ó > <	no es igual a

Finalmente, hay tres operaciones lógicas:

AND **y**
OR **o**
NOT **no**

Estos últimos se usan sobre todo para juntar fórmulas múltiples en instrucciones IF... THEN.

EJEMPLOS:
IF A = B AND C = D THEN 100 exige que tanto A = B como C = D sean ciertos.
IF A = B OR C = D THEN 100 exige que A = B o C = D sean ciertos.

2. COMANDOS

CONT (Continuar)

Este comando se usa para reanudar la ejecución de un programa que haya sido interrumpido pulsando la tecla STOP o como consecuencia de una instrucción STOP o END incorporada en el programa. El programa empezará su ejecución en el punto exacto en el que fue interrumpido.

El comando CONT será invalidado si usted ha cambiado alguna línea del programa o la ha añadido, o si tan sólo ha situado el cursor a otra línea del programa y pulsando RETURN aún cuando no haya cambiado nada. El comando CONT será también invalidado si el paro del programa fué debido a un error, o usted causó un error antes de escribir el comando CONT. En tales casos en la pantalla aparecerá el mensaje: CAN'T CONTINUE ERROR (= No Puedo Continuar. Error).

LIST (listar)

El comando LIST permite hacer un listado de las líneas del programa en BASIC que ha sido grabado en la memoria del ordenador, sea mediante el teclado o desde una cinta cassette o disco floppy. Si usted usa este comando sin indicar números de línea, listará todo el programa en la pantalla (el listado



puede desacelerarse apretando la tecla CTRL o pararse pulsando la tecla RUN/STOP). Si a continuación de LIST usted mecanografía un número de línea, el C 64 mostrará en pantalla sólo dicha línea; si mecanografía dos números de línea separados por un guión, listará todas las líneas comprendidas entre dichos números; si escribe sólo un número de línea seguido de guión, se mostrará todas las líneas a partir de dicho número; finalmente, si escribe un guión y un número de línea, usted obtendrá una lista de todas las líneas hasta ese número. Usando LIST en una u otra de esas variantes, usted puede examinar cualquier parte de un programa o visualizar líneas para proceder a su modificación.

EJEMPLOS:

LIST	Lista todo el programa
LIST 1Ø—	Lista desde la línea 1Ø hasta el final
LIST 1Ø	Lista sólo la línea 1Ø
LIST —1Ø	Lista las líneas desde el principio hasta la n.° 1Ø
LIST 1Ø-2Ø	Lista desde la línea 1Ø a la 2Ø.

LOAD (abreviado L SHIFT O)

Este comando se usa para cargar en la memoria del C 64 un programa almacenado en cinta cassette o en un disco floppy: usted escribe LOAD y pulsa la tecla RETURN y el C 64 buscará el primer programa en la cinta y lo transferirá a la memoria, para su ejecución (con RUN), listado (con LIST), etc. Usted puede también indicar un nombre de programa después de LOAD; este nombre suele entrecomillarse y puede ir seguido de una coma (después de cerrar las comillas) y un número o variable numérica, que señale el dispositivo periférico del que procede el programa. Si no se indica ningún número, el C 64 presupone que se trata del periférico n.° 1, que es el magnetófono cassette.

El otro periférico usado comúnmente con el comando LOAD es la unidad de discos, que tiene asignado el n.° 8.

EJEMPLOS:

LOAD	Carga en memoria el primer programa que encuentra en la cinta.
LOAD"HOLA"	Busca en la cinta hasta encontrar el programa HOLA y lo carga en memoria.
LOAD A$	Busca un programa cuyo nombre está en la variable A$.
LOAD"HOLA",8	Busca el programa HOLA en el diskette
LOAD"*",8	Busca el primer programa del diskette

Además, el comando LOAD puede incorporarse en un programa en BASIC para buscar y ejecutar el siguiente programa de una cinta.

NEW

Este comando borra todo el programa existente en la memoria, así como las variables que puedan haberse usado. A menos que dicho programa se haya almacenado previamente en una cinta o diskette, se pierde irremisiblemente, a no ser que usted lo recuerde y pueda mecanografiarlo de nuevo. Por consiguiente, debe usarse con precaución.



El comando NEW puede incorporarse también en un programa escrito en BASIC: cuando el programa, en su ejecución, llega a esta línea, se borra y todo se para; es útil si usted quiere que todo quede borrado al terminar la ejecución del programa.

RUN

Después de mecanografiar un programa en el teclado o cargarlo con el comando LOAD, se usa el comando RUN para ejecutarlo. Si se escribe sólo RUN, la ejecución del programa empieza en la primera línea; si se indica con un número, se inicia la ejecución a partir de dicha línea.

EJEMPLOS:

RUN	Empieza la ejecución del programa en el número de línea más bajo.
RUN 1ØØ	Empieza el programa en la línea 1ØØ
RUN X	En la pantalla aparece el mensaje "SYNTAX ERROR" (RUN debe escribirse sólo o seguido de un número de línea, no de una letra)

SAVE

Este comando se usa para almacenar el programa existente en la memoria en una cinta o diskette. Si usted escribe sólo SAVE y pulsa RETURN, la máquina intentará almacenar el programa en la cinta del cassette. El Commodore 64 no puede comprobar si ya hay un programa almacenado en dicho punto; por lo que es necesario prestar cuidado con las cintas. Si escribe SAVE seguido de un nombre entrecomillado o una variable tipo cadena, el C 64 adjudicará dicho nombre al programa, para facilitar su localización en el futuro. Después del nombre puede pulsarse una coma (después de cerrar las comillas) y un número o variable numérica. Dicho número indica al C 64 qué dispositivo periférico ha de usar para almacenar el programa: el n.° 1 es el cassette y el n.° 8 la unidad de discos. Puede incluirse también una segunda coma y un número que será 0 ó 1: este "1" sirve para que se ponga una marca "END OF TAPE", que indica el fin de la cinta. Cuando, luego, al querer cargar un programa con LOAD, el C 64 encuentra una de esas marcas, pone en pantalla el siguiente mensaje de error: FILE NOT FOUND ERROR.

EJEMPLO:

SAVE	Almacena el programa en la cinta del cassette sin nombre que lo defina.
SAVE"HOLA"	Almacena en la cinta el programa, con el nombre HOLA.
SAVE A$	Almacena en la cinta el programa, con el nombre contenido en la variable A$.
SAVE"HOLA",8	Almacena en el diskette el programa, con el nombre HOLA.
SAVE"HOLA",1,1	Almacena en la cinta el programa, con el nombre HOLA y seguido de un marcador END OF TAPE.



VERIFY

Este comando ordena al Commodore 64 para que verifique si el programa transferido a cinta o diskette coincide exactamente con la versión en memoria. La finalidad de esta comprobación es asegurarse de que el programa almacenado es correcto y carece de anomalías, tales como una cinta defectuosa. Este comando es muy útil también para situar una cinta de modo que el C 64 pueda grabar otro programa al final del último registrado; para ello, pedir al C 64 que verifique el último programa de la cinta, especificando su nombre; naturalmente, el C 64 informará que los programas no coinciden -lo que usted ya sabía-, pero la cinta entrará en el punto en que puede almacenarse otro programa sin borrar los existentes.

Si se usa VERIFY sin añadir nada al comando, el C 64 comprobará el siguiente programa de la cinta, cualquiera que sea su nombre, comparándolo con el existente en la memoria interna del ordenador. Si después de VERIFY se escribe un nombre de programa (entrecomillado) o una variable de cadena, el C 64 buscará dicho programa y al encontrarlo lo verificará. Si además del nombre se pulsa una coma y un número, el C 64 comprobará el programa en cuestión en el periférico especificado (1 = cinta, 8 = unidad de discos).

EJEMPLO:

VERIFY	Comprueba el siguiente programa de la cinta.
VERIFY"HOLA"	Busca el programa HOLA y lo compara con el de la memoria.
VERIFY"HOLA",8	Busca el programa HOLA en el diskette y lo comprueba.

3. INSTRUCCIONES

CLOSE

Esta instrucción sirve para completar y cerrar los ficheros usados por instrucciones OPEN. Va seguida del número del fichero que se desea cerrar.

EJEMPLO:

CLOSE 2	se cerrará sólo el fichero n.° 2.

CLR

CLR es la abreviación de "clear", que significa borrar; se usa para borrar todas las variables que haya en la pantalla, pero dejando el programa intacto. Esta instrucción es ejecutada automáticamente cuando se escribe un comando RUN.

CMD

CMD envia a otro periférico la información de salida, que de otro modo sería transferida a la pantalla (por ejemplo, con instrucciones PRINT o LIST, pero no POKE). Dicho periférico puede ser una impresora o un fichero de datos en cinta o diskette, y ha de abrirse previamente con una instrucción OPEN. CMD va seguida de un número o variable numérica, para indicar el fichero.



EJEMPLO:

OPEN 1,4 Abre el periférico n.º 4, que es la impresora.

CMD 1 Toda la salida normal irá ahora a la impresora.

LIST El listado va a la impresora, en vez de a la pantalla, (incluso la palabra LIST acabada de escribir será visualizada en la impresora).

Para volver la salida a la pantalla, cerrar simplemente el fichero con el comando CLOSE.

DATA

La instrucción DATA va seguida de una serie de datos a usar en instrucciones READ. Los datos pueden ser números o palabras, separados por comas. Las palabras no precisan estar entre comillas, a menos que contengan alguno de los siguientes caracteres: ESPACIO, dos puntos o coma. Si se escriben dos comas seguidas, sin nada entre ellas, el valor será interpretado por la instrucción READ como un cero (si se trata de números), o como una cadena vacía.

EJEMPLO DE INSTRUCCION DATA:

```
DATA 100,200,FRED,"HOLA,MADRE",,3.14,abc123
```

Puesto que el programa no necesita ejecutar una instrucción DATA para leer la información, es conveniente situar las instrucciones DATA lo más cerca posible de la última línea del programa. De esta forma el programa es ejecutado más rápidamente.

DEF FN (Define Función)

Esta instrucción permite definir un cálculo completo como una función designada con un nombre corto. Esto ahorra considerable espacio de memoria cuando una fórmula larga debe usarse varias veces en el curso de un mismo programa.

El nombre que usted puede dar a la función en las letras FN seguidas de cualquier nombre legal de variable (de 1 ó 2 caracteres). Primero, usted define la función usando la instrucción DEF seguida del nombre que adjudica a la función; a continuación escriba una variable numérica -en este caso X- entre paréntesis y finalmente el signo igual a (=) y la fórmula en cuestión. De este modo usted puede volver a utilizar la fórmula sustituyendo la X por cualquier número, como se muestra en la línea 20 del siguiente ejemplo:

EJEMPLO:

```
1Ø DEF FNA(X) = 12*(34.75-X/.3)
2Ø PRINT FNA(7)
```

El n.º se inserta en el lugar en que había una X en la fórmula

DIM (Dimensión de una tabla)

Antes de poder usar tablas de variables, a menos que el número de elementos no exceda de 11, el programa debe ejecutar una instrucción DIM para la tabla en cuestión. A continuación de DIM se escribe el nombre de la tabla, que puede consistir en el nombre de cualquier variable legal; luego, entre paréntesis, se especifica el número (o variable numérica) de elementos en cada dimensión. Una tabla que tenga más de una dimensión se denomina "matriz". Usted puede usar cualquier número de dimensiones, pero tenga en cuenta que la lista de variables que está creando ocupa gran cantidad de memoria y es fácil agotar ésta en este tipo de operaciones. Para obtener el número de variables creadas con cada instrucción DIM, multiplique el número total de elementos en cada dimensión de la matriz.



EJEMPLO:

```
10 DIM A$(40), B7(15), CC%(4,4,4)
          ↑         ↑           ↑
41 Elementos  16 Elementos  125 Elementos
```

En una misma instrucción DIM es posible dimensionar más de una tabla, separando éstos por comas. Hay que prestar atención a que el programa no ejecute instrucciones DIM más de una vez para una misma tabla, o de lo contrario se recibirá un mensaje de error en la pantalla. Es conveniente situar las instrucciones DIM cerca del principio del programa.

END

Cuando un programa en su ejecución llega a una instrucción END, queda parado, como si se hubieran agotado sus líneas. Para reanudar la ejecución, usar el comando CONT.

FOR..TO...STEP

Esta instrucción se usa en combinación con la instrucción NEXT para establecer una sección de programa que se repita un número dado de veces. Por ejemplo, usted puede hacer que el C 64 cuente hasta un número elevado para intercalar una pausa de unos segundos dentro de un programa, o que cuente, de ser necesario, alguna cosa. Estas instrucciones se hallan entre las más usadas del lenguaje BASIC.

La instrucción se escribe con el siguiente formato:

FOR (nombre de la variable de bucle) = (principio de contaje) TO (final de contaje); donde la variable de bucle es la variable a cuyo valor se irá sumando o restando durante la ejecución del programa. El comienzo y el final de contaje son los límites del valor adjudicado a la variable de bucle.

La instrucción FOR tiene la siguiente lógica: primero, la variable bucle se ajusta al valor de contaje inicial; el valor de contaje final es almacenado para posterior referencia para el C 64. Cuando el programa en su ejecución llega a una línea que contiene un comando NEXT, añade 1 al valor de la variable bucle y comprueba si es mayor que el valor final del bucle. Si no es mayor, se ejecuta la línea que sigue inmediatamente a la instrucción FOR. Si, en cambio, la variable de bucle es mayor que el número final del bucle, se ejecutará la línea siguiente a la instrucción NEXT.

EJEMPLO:

```
1Ø FOR L = 1 TO 1Ø
2Ø PRINT L
3Ø NEXT L
4Ø PRINT "ESTOY FRITO!L = "L
```




Este programa pondrá en pantalla los números del 1 al 10, seguidos del mensaje ESTOY FRITO!L = 11. ¿Comprende usted cómo opera? En caso negativo, vuelva a leer el párrafo que precede al ejemplo y anote en un papel cada paso del programa.

El valor final del bucle puede ir seguido de la palabra STEP y otro número o variable. En tal caso, cada vez se añadira el valor que sigue a STEP, en vez de 1. Permite hacer cuentas hacia atrás o por fracciones o de cualquier otra forma deseada.

Usted puede establecer bucles dentro de otros. El bucle interior se denomina "encajado". Hay que prestar atención a que el último bucle en empezar sea el primero en terminar.

EJEMPLO DE BUCLES ENCAJADOS:

```
1Ø FOR L=1 TO 1ØØ
2Ø FOR A=5 TO 11 STEP 2    ←——  "for... next" está
3Ø NEXT A                       encajado dentro
4Ø NEXT L                       del otro mayor.
```

```
Incorrecto:
1Ø FOR L=1 TO 1ØØ
2Ø FOR A=5 TO 11 STEP 2
3Ø NEXT L
4Ø NEXT A
```

GET

La instrucción GET es una forma de obtener datos del teclado uno a uno. Cuando se ejecuta GET, el carácter pulsado es recibido; si no se pulsa ninguno, se recibe un carácter cero (vacío) y el programa continúa. No hay necesidad de pulsar la tecla RETURN.

GET se escribe seguido de un montaje de variable, generalmente una variable tipo cadena (alfanumérica). Si se usara una variable numérica y luego se pulsara una tecla que no fuera un número, el programa se pararía con un error. La instrucción GET puede introducirse también en un bucle que compruebe un resultado vacío, de modo que el programa espere la pulsación de una tecla.

EJEMPLO:

```
1Ø GET A$: IF A$="''THEN 1Ø   ←——  Esta línea espera
                                   que se pulse una
                                   tecla. Al pulsar
                                   cualquier tecla el
                                   programa continuará.
```

GET

Se usa con un periférico previamente abierto con el comando OPEN, a fin de entrar caracteres de uno en uno.

EJEMPLO:

```
GET #1,A$
```




GOSUB (Go Subroutine)

Esta instrucción opera como la GOTO, excepto en que el C 64 recuerda de dónde vino: cuando se encuentra una línea que contiene una instrucción RETURN, el programa retrocede a la instrucción que seguía a GOSUB. Es útil si en el programa hay una rutina que ocurre varias veces en diferentes partes del programa; en vez de escribir de nuevo la rutina cada vez, usted la mecanografía una sola vez y luego vuelve a ella utilizando la instrucción GOSUB en cualquier otra parte del programa. Así, la línea "20 GOSUB 800" significa "ir a la subrutina que comienza en la línea 800 y ejecutarla".

GOTO o GO TO

Cuando, en su ejecución, el programa llega a una instrucción GOTO, el programa pasa a ejecutar la línea indicada después de la palabra GOTO.

IF...THEN

Esta instrucción permite al C 64 analizar una situación y adoptar dos diferentes decisiones según el resultado de dicho análisis. Así, si la expresión evaluada resulta verdadera, se ejecuta la instrucción que sigue a THEN; dicha instrucción puede ser un número de línea, en cuyo caso el C 64 irá a dicha línea del programa, o puede ser cualquier otra instrucción o instrucciones BASIC. Si la expresión resulta falsa, se ejecuta la siguiente línea, en vez de la siguiente instrucción en la misma línea.

La expresión evaluada puede ser una variable o una fórmula, en cuyo caso se considera verdadera si no es cero y falsa si es cero. En la mayoría de los casos hay una expresión que incluye operadores relacionales (= ,,, = , = ,, AND, OR, NOT). Si el resultado es verdadero, tiene el valor de -1, y si es falso, el valor 0. Véase la sección sobre operadores relacionales, en que se explica su operación.

INPUT

La instrucción INPUT permite mecanografiar datos en el teclado que son asignados por el ordenador a una variable. A este fin, el programa interrumpe su ejecución, visualiza un interrogante (?) en la pantalla y espera que el ordenador escriba la respuesta y pulse la tecla RETURN.

Al componer el programa, INPUT va seguido del nombre de una variable o de una lista de nombres de variables separados por comas. De desearse, antes de la lista de variables puede incluirse un mensaje entre comillas; este mensaje se denomina "aviso" y va seguido de punto y coma(;) después de cerrar las comillas. Cuando en una instrucción INPUT hay más de una variable, las respuestas se mecanografían separadas por comas.

EJEMPLO:

```
1Ø INPUT"ESCRIBA A #";A
2Ø INPUT"Y SU NOMBRE";A$
3Ø INPUT B$
4Ø PRINT"SEGURO QUE NO SABIA LO QUE QUERIA"
```




INPUT #

Esta instrucción opera como INPUT, pero extrae los datos desde un fichero o periférico previamente abierto con un comando OPEN.

LET

La instrucción LET es frecuentemente usada en todos los programas escritos en BASIC, pero puede prescindirse de su designación, siendo opcional el escribir la palabra LET. El nombre de la variable a la que se asigna el resultado de un cálculo se pone a la izquierda del signo igual a (=) y el número o fórmula se sitúa a su izquierda.

EJEMPLO:

```
1Ø LET A=5
2Ø B=6
3Ø C=A*B+3
4Ø D$="HOLA"
```

NEXT

NEXT se usa siempre asociada a la instrucción FOR. Cuando el programa llega a la instrucción NEXT, vuelve a la línea que contiene FOR y comprueba el bucle (véase la instrucción FOR para más detalles). Si el bucle está terminado, la ejecución pasa a la instrucción que sigue a NEXT. NEXT puede ir seguido de un nombre de variables o una lista de nombres de variables separados por comas. Si no se indica ninguna variable, se completa el último bucle empezado; si hay variables, éstas se completan en el orden en que se introdujeron, es decir de izquierda a derecha.

EJEMPLO:

```
1Ø FOR L=1 TO 1Ø:NEXT
2Ø FOR L=1 TO 1Ø:NEXT L
3Ø FOR L=1 TO 1Ø:FOR M=1 TO 10:NEXT M,L
```

ON

Esta instrucción puede convertir GOTO o GOSUB en versiones especiales de la instrucción IF. ON va seguido de una fórmula, que se evalúa para obtener un número; GOTO o GOSUB van seguidas con una lista de número de línea separados por comas. Si el resultado del cálculo de la fórmula es 1, se ejecuta la línea correspondiente al primer número de la lista; si el resultado es 2, se ejecuta el segundo número de línea, y así sucesivamente. Si el resultado es 0 o bien negativo o mayor que la lista de números de línea, se ejecuta la instrucción siguiente a ON.

EJEMPLO:

```
1Ø INPUT X
2Ø ON X GOTO 1Ø,5Ø,5Ø,5Ø
3Ø PRINT "QUE VA!"
4Ø GOTO 1Ø
5Ø PRINT "VIVA!"
6Ø ON X GOTO 1Ø,3Ø,3Ø
```




OPEN

La instrucción OPEN permite tener acceso a periféricos, tales como un cassette, o una unidad de discos, una impresora o incluso la propia pantalla utilizada por el C 64. OPEN va seguido de un número de referencia, a usar con el resto de instrucciones BASIC, y que puede seleccionarse entre 1 y 255. Normalmente se indica un segundo número, separado por una coma, para especificar el periférico: 0 = pantalla, 1 = cassette, 4 = impresora, 8 = unidad de discos. Puede añadirse un tercer número, también separado por una coma como dirección secundaria, en el caso del cassette se usa el 0 para transferir datos de la cinta al ordenador; 1 para grabar datos en la cinta y 2 para grabación incluyendo un marcador de final de cinta. Cuando se usa la unidad de discos, este tercer número indica la memoria buffer o canal. Con la impresora, el tercer número puede usarse como comando seleccionable (véase el Manual de Referencia del Programador). Finalmente puede escribirse una cadena, consistente en un comando para la manipulación de la unidad de discos o el nombre del fichero de la cinta.

EJEMPLOS:

1Ø OPEN 1,Ø	Abre la pantalla como periférico.
2Ø OPEN 2,1,Ø,"D"	Abre el cassette para extraer datos del fichero denominado "D".
3Ø OPEN 3,4	Para usar la impresora.
4Ø OPEN 4,8,15	Abre el canal de datos en el diskette.

(Véase también: instrucciones CLOSE, CMD, GET #, INPUT # y PRINT #; variable ST, y Apéndice B).

POKE

La instrucción POKE va seguida siempre de dos números o fórmulas. El primer número es una posición en la memoria interna del C 64. Las posiciones están numeradas desde 0 a 65000. Algunos pueden usarse fácilmente en los programas, como las descritas sobre sonido y color, mientras otras son para uso exclusivo del C 64. El experimentar con instrucciones POKE a veces produce resultados curiosos; si ocurre algo y usted no puede pararlo, desconecte el ordenador y vuelva a conectarlo, o apriete RUN/STOP y pulse RESTORE.

El segundo número puede seleccionarse entre 0 y 255. Se grabará en la posición de memoria especificada, sustituyendo a cualquier valor que hubiera en ella alojado.

EJEMPLO:

```
1Ø POKE 36879,8
2Ø POKE 9↑16↑3+15,27
```

PRINT

PRINT es una de las primeras instrucciones en aprenderse, pero no por ello deja de tener sus sutilezas. Puede complementarse con los siguientes elementos:



Palabras entre comillas
Nombres de variables
Funciones
Signos de puntuación.

Las palabras entrecomillas se llaman a veces "literales" porque se muestran en pantalla literalmente tal como se mecanografían. Los nombres de variables situados fuera del área entrecomillada se representan en la pantalla con el valor que contienen. También las funciones que se especifican con PRINT son visualizadas con el valor que se les ha adjudicado. Los signos de puntuación se emplean para conferir a los datos un formato nítido: la coma divide la pantalla en 2 columnas equivalentes, mientras que punto y coma no deja espacio en medio. Ambos signos pueden usarse como el último símbolo de la instrucción; en tal caso el siguiente elemento visualizado con PRINT aparece como si continuara la misma instrucción.

EJEMPLO:
```
1Ø PRINT "HOLA"
2Ø PRINT "HOLA,"A$
3Ø PRINT A + B;
5Ø PRINT J;
   PRINT A,B,C,D,
```
Véanse también las funciones: POS(), SPC(), TAB().

PRINT #

Hay algunas diferencias entre esta instrucción y la anterior. En primer lugar, PRINT # va seguido de un número que indica el periférico o fichero de datos previamente abierto con la instrucción OPEN. Dicho número va seguido de una coma y una lista de caracteres que serán escritos. El uso de coma y de punto y coma es idéntico al explicado con la instrucción PRINT, si bien algunos periféricos no operan con las funciones TAB y SPC.

EJEMPLO:
```
1ØØ PRINT #,"HOLA, QUE TAL!";A$,B$
```

READ

Esta instrucción se usa para dar la información de instrucciones DATA a variables, en las que se puede usar. Hay que evitar el uso de READ con cadenas en instrucciones, cuando el comando espera leer un número, ya que en tales casos aparecera el mensaje de error; TYPE MISMATCH ERROR.

REM (remark)

REM introduce simplemente observaciones para la persona que lista el programa. En esta observación puede explicarse una sección del programa, indicarse el nombre del programador, etc. Las instrucciones REM no afectan la ejecución del programa, si bien incrementan su longitud. REM puede ir seguida de cualquier texto, sin embargo el uso de caracteres gráficos dará extraños resultados (véase el Manual de Referencia del Programador).



RESTORE

Cuando esta instrucción se incorpora en un programa, el puntero que señala el siguiente elemento a leer en una instrucción DATA se desplaza al primer elemento de la lista. Esto brinda la oportunidad de volver a leer los datos. La palabra RESTORE se escribe sola en la línea.

RETURN

Esta instrucción se incorpora en programas siempre con GOSUB: cuando el programa llega a RETURN, pasa a la instrucción que sigue a GOSUB. Si no existe una instrucción GOSUB previa, aparecerá el mensaje: RETURN WITHOUT GOSUB ERROR. RETURN no va seguido de ningún elemento complementario.

STOP

Esta instrucción se incorpora en un programa para pararlo. En la pantalla aparece el mensaje "BREAK ERROR IN LINE xxxx", donde xxxx es el número de línea en que se halla STOP. El programa puede reanudarse escribiendo el comando CONT. La instrucción STOP se usa para eliminar errores de un programa.

SYS

La palabra SYS va seguida de un número decimal o de una variable numérica en la gama 0-65535. Con ello, el programa empezará a ejecutar el programa en lenguaje máquina a partir de la posición de memoria especificada. Es similar a la función USR, pero no permite el paso de parámetros.

WAIT

La instrucción WAIT se usa para interrumpir la ejecución del programa hasta que el contenido de una posición de memoria se haya cambiado de una forma especifica. Para ello, la palabra WAIT se sigue con el número de dicha posición de memoria, una coma y otro número. De desearse, puede escribirse un tercer número, también separado por una coma. Los dos últimos números deben estar comprendidos entre 0 y 255.

El contenido de la posición de memoria es primero evaluado con el operador exclusivo OR utilizando el tercer número, de haberlo, y luego lógicamente con el operador AND utilizando el segundo número. Si el resultado es cero, el programa vuelve a la posición de memoria en cuestión para repetir la comprobación; cuando el resultado no es cero, el programa pasa a ejecutar la siguiente instrucción.

4. FUNCIONES

a. NUMERICAS

ABS(X) (valor absoluto)

Esta función da el valor del número sin su signo (— ó +). La respuesta es, pues, siempre positiva.



ATN(X) (arco-tangente)

Da el ángulo, medido en radianes, cuya tangente es X.

COS(X) (coseno)

Da el valor del coseno de X, donde X es un ángulo medido en radianes.

EXP(X) (exponenciación)

Da el valor de la constante matemática "e" (2.71827183) elevado a la potencia de X.

FNXX(X) (función XX)

Da el valor de la función XX definida por el usuario con una instrucción DEFF FNXX.

INT(X) (enteros)

Da el valor de X, es decir omitiendo los decimales. El valor es siempre inferior o igual a X. Cualquier número negativo con decimales se convierte en el número entero inferior a su valor actual.

Si la función INT se usa para redondear al entero más próximo, la forma es: INT(X 2 .5).

EJEMPLO:

X = INT(X*1ØØ + .5)/1ØØ	Redondear al céntimo o centésimo más próximo.

LOG(X) (logaritmo)

Da el valor logarítmico natural de X, referido a la base e (véase la función EXP(X)). Para convertir a la base logarítmica 10, dividir simplemente por LOG(1Ø).

PEEK(X)

Se usa para averiguar el contenido de la posición de memoria X, en la gama de 0 a 65535, y da un resultado entre 0 y 255. Esta función se usa a menudo combinada con la instrucción POKE.

RDN(X) (número aleatorio)

Esta función dará un número aleatorio, es decir al azar, entre 0 y 1. Es útil en juegos, para simular dados y otros elementos casuales, así como en ciertas aplicaciones estadísticas. El primer número aleatorio debería ser generado por la fórmula RND(-TI), para empezar cada vez de modo distinto. El número a colocar en el lugar de X debe ser 1 o cualquier número positivo. Si X fuera cero, el número aleatorio sería el mismo que la última vez. Un valor negativo de X actúa sobre el generador de números de modo que produciría la misma secuencia de números aleatorios.



Para simular un juego de dados, úsese la fórmula INT(RND(1)*6 + 1). Primero, el número aleatorio entre 0 y 1 es multiplicado por 6, lo que extiende la gama a 0-6 (en realidad, mayor que cero y menor de seis). Luego se añade, 1 convirtiendo la gama entre 1 y 7. La función INT suprime los decimales, haciendo que el resultado sea un dígito entre 1 y 6.

Para simular los resultados de 2 dados, sumar dos de los números obtenidos por la anterior fórmula.

EJEMPLO:

1ØØ X = INT(RND(1)*6) + INT(RND(1)*6) + 2	Simula 2 dados.
1ØØ X = INT(RND(1)*1ØØØ) + 1	Número entre 1 y 1ØØØ.
1ØØ X + INT(RND(1)*15Ø) + 1ØØ	Número entre 1ØØ y 249.

SGN(X) (signo)

Esta función da el signo de X, sea positivo, negativo o cero. El resultado será + 1 de ser positivo, 0 de ser cero y -1 de ser negativo.

SIN(X) (seno)

Función de seno trigonométrico. El resultado es el seno de X, donde X es un ángulo expresado en radianes.

SQR(X) (square root = raíz cuadrada)

Esta función da la raíz cuadrada de X, donde X es un número positivo o cero. Si X es un número negativo, en la pantalla aparece un mensaje "ILLEGALY QUANTITY ERROR".

TAN(X) (tangente)

El resultado es la tangente de X, donde X es un ángulo en radianes.

USR(X)

Cuando se usa esta función, el programa salta a un programa en lenguaje máquina, cuyo punto de partida está en las posiciones de memoria 1 y 2. El parámetro X se pasa al programa en lenguaje de máquina, el cual dará como respuesta otro número para el programa en BASIC. Véase el Manual de Referencia del Programador, en que se trata más detalladamente esta función y la programación en lenguaje de máquina.

b. FUNCIONES DE CADENA

ASC(X$)

Esta función da el código ASCII del primer carácter de la variable X$.



CHR$(X)

Esta función es la opuesta a ASC, y produce un carácter de cadena cuyo código ASCII es X.

LEFT$(X$,X)

Esta función da una cadena que contiene los caracteres X más a la izquierda de la variable X$.

LEN(X$)

Esta función da el número de caracteres (incluyendo espacios y otros símbolos) de la cadena X$.

MID$ (X$,S,X)

Esta función da una cadena que contiene los caracteres X, empezando por *el carácter número S, de la variable X$.*

RIGHT$(X$,X)

Esta función da los caracteres X más a la derecha de la variable X$.

STR$(X)

Contendrá una cadena idéntica a la versión de X$ visualizada por la instrucción PRINT.

VAL(X$) (Valor)

Esta función convierte la cadena X$ en un número y es esencialmente la operación opuesta a STR$. La cadena es examinada desde el carácter más a la izquierda, hasta donde llega el formato reconocible de los números. Si el Commodore 64 encuentra caracteres ilegales, se convierte sólo la parte de cadena hasta dicho punto.

EJEMPLO:

1Ø X ≐ VAL("123.456)	X = 123.456
1Ø X = VAL("12A13B")	X = 12
1Ø X = VAL("RIUO17*")	X = Ø
1Ø X = VAL("—1.23.23.23")	X = —1.23

c. OTRAS FUNCIONES

FRE(X) (libre)

Esta función da el número de bytes no usados disponibles en la memoria, cualquiera que sea el valor de X.



Puede ser que esta función le devuelva un número negativo debido a que el almacenamiento del valor de los bytes libres se hace en una variable entera (rango: de +32767 a —32768). Para solventar este problema, sume a la cantidad negativa el número 65536, lo cual le dará el número real de bytes libres en la memoria del ordenador.

NOTA: sólo sumar en el caso de que devuelva un número negativo; en caso contrario, es ESE el valor de bytes libres.

POS(X) (posición)

Esta función da el número de la columna (0-21) en que empezará la siguiente instrucción PRINT en la pantalla. X puede tener cualquier valor, y no se usa.

SPC(X) (espacio)

Se usa en la instrucción PRINT para saltar X espacios.

TAB(X) (tabulación)

Esta función se usa en instrucciones PRINT, para visualizar el siguiente carácter en la columna número X.



APENDICE D:

ABREVIACIONES DE LAS PALABRAS CLAVE DE BASIC:

Para ayudarle a ganar tiempo a la hora de mecanografiar los programas y comandos, el BASIC del Commodore-64 le permite abreviar la mayoría de las palabras claves. La abreviación para "PRINT" es un interrogante. Las abreviaciones para otras palabras se consiguen escribiendo la primera o las dos primeras letras de la palabra, y a continuación la letra siguiente de la palabra con SHIFT (o sea pulsando simultáneamente SHIFT y la letra). Si estas abreviaciones se utilizan en un programa, cuando sea listado aparecerá la palabra BASIC entera (sin abreviar). Observe que algunas de las palabras, cuando están abreviadas, incluyen un paréntesis izquierdo.

Palabra	Abreviación	Símbolo en la pantalla
ABS	A SHIFT B	A
AND	A SHIFT H	A
ASC	A SHIFT S	A ♥
ATN	A SHIFT T	A
CHR$	C SHIFT H	C
CLOSE	CL SHIFT O	CL
CLR	C SHIFT L	C
CMD	C SHIFT M	C
CONT	C SHIFT O	C
DATA	D SHIFT A	D ♠
DEF	D SHIFT E	D
DIM	D SHIFT I	D
END	E SHIFT N	E
EXP	E SHIFT X	E ♣

Palabra	Abreviación	Símbolo en la pantalla
FOR	F SHIFT O	F
FRE	F SHIFT R	F
GET	G SHIFT E	G
GOSUB	GO SHIFT S	GO ♥
GOTO	G SHIFT O	G
INPUT#	I SHIFT N	I
LET	L SHIFT E	L
LEFT$	LE SHIFT F	LE
LIST	L SHIFT I	L
LOAD	L SHIFT O	L
MID$	M SHIFT I	M
NEXT	N SHIFT E	N
NOT	N SHIFT O	N
OPEN	O SHIFT P	O

Palabra	Abreviación	Símbolo en la pantalla
PEEK	P SHIFT E	P
POKE	P SHIFT O	P
PRINT	?	?
PRINT#	P SHIFT R	P
READ	R SHIFT E	R
RESTORE	RE SHIFT S	RE ♥
RETURN	RE SHIFT T	RE
RIGHT$	R SHIFT I	R
RND	R SHIFT N	R
RUN	R SHIFT U	R
SAVE	S SHIFT A	S ♠
SGN	S SHIFT G	S
SIN	S SHIFT I	S

Palabra	Abreviación	Símbolo en la pantalla
SPC(	S SHIFT P	S
SQR	S SHIFT Q	S ●
STEP	ST SHIFT E	ST
STOP	S SHIFT T	S
STR$	ST SHIFT R	ST
SYS	S SHIFT Y	S
TAB	T SHIFT A	T ♠
THEN	T SHIFT H	T
USR	U SHIFT S	U ♥
VAL	V SHIFT A	V ♠
VERIFY	V SHIFT E	V
WAIT	W SHIFT A	W ♠

CODIGOS DE PANTALLA

La siguiente tabla, contiene todos los caracteres que posee el Commodore 64 en su juego de caracteres. Muestra que número deberá entrar por POKE en la memoria de pantalla (registro 1024 a 2023) para obtener el caracter deseado. También se puede ver, a que caracter le corresponde el número que se ha entrado por PEEK.

Hay dos juegos de caracteres disponibles pero sólo uno a la vez esto quiere decir que no puede tener simultáneamente en pantalla caracteres de uno y otro juego. Se puede cambiar de juego pulsando simultáneamente las teclas SHIFT y COMMODORE.

Desde BASIC, haga POKE 53272,21 para obtener las mayúsculas y POKE 53272,23 para cambiar a minúsculas.

Cualquier caracter de la tabla puede ser presentado en modo INVERSO.

El código del caracter en INVERSO se obtiene, añadiendo al código del caracter "128".

Si quiere visualizar un círculo en la situación 1504, haga un POKE con el código del círculo (81) en el registro 1504: POKE 1504,81.

Hay también una situación de memoria correspondiente para el control del color de cada caracter presentado en pantalla (registros 55296-56295). Para cambiar el color del círculo a amarillo (código de color 7), hará un POKE en la situación de memoria correspondiente (55776) con el código: POKE 55776,7.

Refiérase al apéndice G para los mapas de memoria de pantalla y color, con sus códigos correspondientes.

CODIGOS DE PANTALLA

JUEGO 1	JUEGO 2	POKE	JUEGO 1	JUEGO 2	POKE	JUEGO 1	JUEGO 2	POKE
@		0	C	c	3	F	f	6
A	a	1	D	d	4	G	g	7
B	b	2	E	e	5	H	h	8

JUEGO 1	JUEGO 2	POKE	JUEGO 1	JUEGO 2	POKE	JUEGO 1	JUEGO 2	POKE
I	i	9	%		37	♠	A	65
J	j	10	&		38		B	66
K	k	11	'		39		C	67
L	l	12	(		40		D	68
M	m	13	)		41		E	69
N	n	14	*		42		F	70
O	o	15	+		43		G	71
P	p	16	,		44		H	72
Q	q	17	–		45		I	73
R	r	18	.		46		J	74
S	s	19	/		47		K	75
T	t	20	0		48		L	76
U	u	21	1		49		M	77
V	v	22	2		50		N	78
W	w	23	3		51		O	79
X	x	24	4		52		P	80
Y	y	25	5		53	●	Q	81
Z	z	26	6		54		R	82
[		27	7		55	♥	S	83
£		28	8		56		T	84
]		29	9		57		U	85
↑		30	:		58	╳	V	86
←		31	;		59	○	W	87
SPACE		32	<		60	♣	X	88
!		33	=		61		Y	89
"		34	>		62	♦	Z	90
#		35	?		63	┼		91
$		36			64	▒		92



JUEGO 1	JUEGO 2	POKE	JUEGO 1	JUEGO 2	POKE	JUEGO 1	JUEGO 2	POKE
		93			105			117
		94			106			118
		95			107			119
SPACE		96			108			120
		97			109			121
		98			110			122
		99			111			123
		100			112			124
		101			113			125
		102			114			126
		103			115			127
		104			116			

Los códigos de 128 a 255 representan los mismos caracteres que los códigos de 0 a 127, pero en modo INVERSO.

APENDICE F:

CODIGOS ASCII y CHR$

Este Apéndice le muestran qué caracteres aparecen si escribe: PRINT CHR$(X), para todos los valores posibles de X. También le muestra los valores que obtendrá si escribe PRINT ASC ("x"), donde x es cualquier carácter que pueda escribir. Esto es muy práctico para comparar un caracter recibido por una instrucción GET, para convertir la presentación de Mayúsculas a Minúsculas y escribir comandos (basados en caracteres (como el conmutador Mayúsculas/Minúsculas) que no podrían ser escritos dentro de las comillas de un PRINT.

VISUALIZAR	CHR$	VISUALIZAR	CHR$	VISUALIZAR	CHR$	VISUALIZAR	CHR$
	0	CRSR	17	"	34	3	51
	1	RVS ON	18	#	35	4	52
	2	CLR HOME	19	$	36	5	53
	3	INST DEL	20	%	37	6	54
	4		21	&	38	7	55
WHT	5		22	'	39	8	56
	6		23	(	40	9	57
	7		24	)	41	:	58
ANULAR SHIFT C=	8		25	*	42	;	59
PONER SHIFT C=	9		26	+	43	<	60
	10		27	,	44	=	61
	11	RED	28	−	45	>	62
	12	CRSR →	29	.	46	?	63
RETURN	13	GRN	30	/	47	@	64
CONMUTADOR MINUSCULAS	14	BLU	31	0	48	A	65
	15	SPACE	32	1	49	B	66
	16	!	33	2	50	C	67

VISUALIZAR	CHR$	VISUALIZAR	CHR$	VISUALIZAR	CHR$	VISUALIZAR	CHR$
D	68		97		126		155
E	69		98		127	PUR	156
F	70		99		128	CRSR	157
G	71		100		129	YE	158
H	72		101		130	CYN	159
I	73		102		131	SPACE	160
J	74		103		132		161
K	75		104	f1	133		162
L	76		105	f3	134		163
M	77		106	f5	135		164
N	78		107	f7	136		165
O	79		108	f2	137		166
P	80		109	f4	138		167
Q	81		110	f6	139		168
R	82		111	f8	140		169
S	83		112	SHIFT RETURN	141		170
T	84		113	CONMUTADOR MAYUSCULAS	142		171
U	85		114		143		172
V	86		115	BLK	144		173
W	87		116	CRSR	145		174
X	88		117	RVS OFF	146		175
Y	89		118	CLR HOME	147		176
Z	90		119	INST DEL	148		177
[	91		120		149		178
£	92		121		150		179
]	93		122		151		180
↑	94		123		152		181
←	95		124		153		182
	96		125		154		183

VISUALIZAR	CHR$	VISUALIZAR	CHR$	VISUALIZAR	CHR$	VISUALIZAR	CHR$
	184		186		188		190
	185		187		189		191

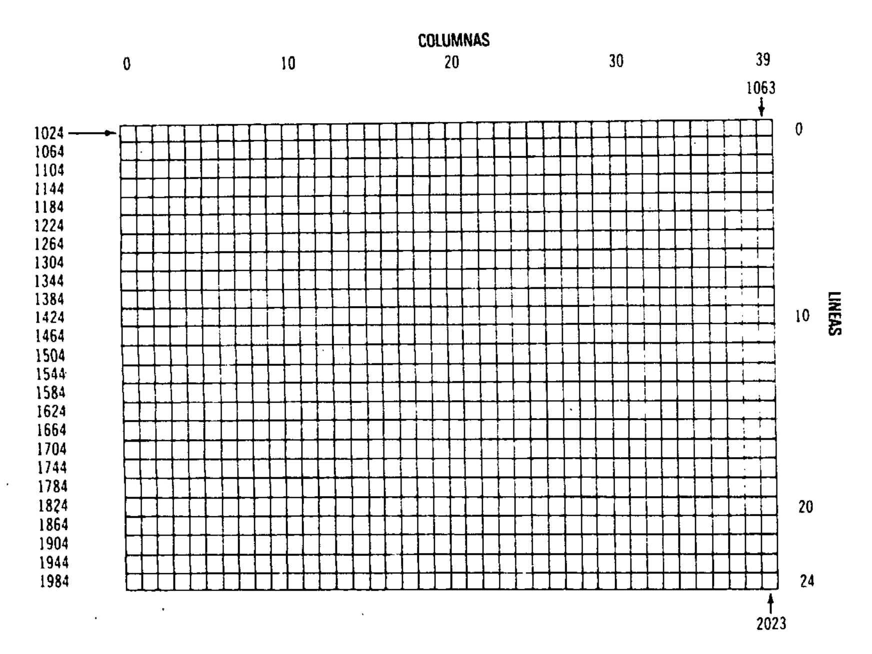
LOS CODIGOS	192 A 223	SON LOS MISMOS QUE	96 A 127
LOS CODIGOS	224 A 254	SON LOS MISMOS QUE	160 A 190
EL CODIGO	255	ES EL MISMO QUE	126

APENDICE G:

LOS MAPAS DE MEMORIA DE COLOR Y PANTALLA

Las tablas siguientes muestran, que situación de memoria controla el emplazamiento de los caracteres en la pantalla, y el registro que se utiliza para cambiar individualmente el color de un caracter, así como la lista de los códigos de colores.

MAPA DE LA MEMORIA DE PANTALLA

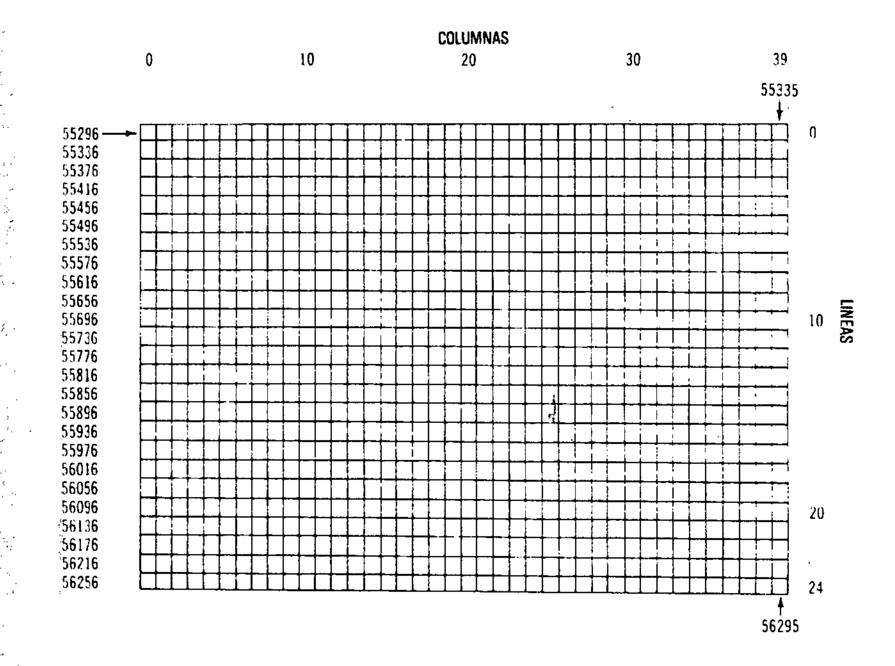


Los valores de los códigos de color que se pueden entrar por POKE, en un registro para cambiar el color de un caracter son los siguientes:

Ø	NEGRO	8	NARANJA
1	BLANCO	9	MARRON
2	ROJO	1Ø	ROJO Claro
3	CIAN	11	GRIS 1
4	PURPURA	12	GRIS 2
5	VERDE	13	VERDE Claro
6	AZUL	14	AZUL Claro
7	AMARILLO	15	GRIS 3

Por ejemplo para cambiar el color de un caracter situado en la esquina izquierda superior de la pantalla, al rjo; escriba: POKE 55296,2.

MAPA DE LA MEMORIA DE COLORES





APENDICE H:

DERIVACION DE FUNCIONES MATEMATICAS

Las funciones que no son intrínsecas al BASIC del Commodore 64 pueden calcularse de la forma siguiente:

FUNCION	EQUIVALENTE EN BASIC
SECANTE	SEC(X) = 1/COS(X)
COSECANTE	CSC(X) = 1/SIN(X)
COTANGENTE	COT(X) = 1/TAN(X)
SENO INVERSO	ARCSIN(X) = ATN(X/SQR(—X*X + 1))
COSENO INVERSO	ARCCOS(X) = ATN(X/SQR (—X*X + 1)) + π/2
SECANTE INVERSA	ARCSEC(X) = ATN(X/SQR(X*X—1))
COSECANTE INVERSA	ARCCSC(X) = ATN(X/SQR(X*X—1)) + (SGN(X)—1*π/2
COTANGENTE INVERSA	ARCOT(X) = ATN(X) + π/2
SENO HIPERBOLICO	SINH(X) = (EXP(X)—EXP(—X))/2
COSENO HIPERBOLICO	COSH(X) = (EXP(X) + EXP(—X))/2
TANGENTE HIPERBOLICA	TANH(X) = EXP(—X)/(EXP(x) + EXP (—X))*2 + 1
SECANTE HIPERBOLICA	SECH(X) = 2/(EXP(X) + EXP(—X))
COSECANTE HIPERBOLICA	CSCH(X) = 2/(EXP(X)—EXP(—X))
COTANGENTE HIPERBOLICA	COTH(X) = EXP(—X)/(EXP(X) —EXP(—X))*2 + 1
SENO HIPERBOLICO INVERSO	ARCSINH(X) = LOG(X + SQR(X*X + 1))
COSENO HIPERBOLICO INVERSO	ARCCOSH(X) = LOG(X + SQR(X*X—1))
TANGENTE HIPERBOLICA INVERSA	ARCTANCH(X) = LOG((1 + X)/(1—X))/2
SECANTE HIPERBOLICO INVERSO	ARCSECH(X) = LOG((SQR (—X*X + 1) + 1/X)
COSECANTE HIPERBOLICA INVERSA	ARCCSCH(X) = LOG((SGN(X)*SQR (X*X + 1/x)
COTANGENTE HIPERBOLICA INVERSA	ARCCOTH(X) = LOG((X + 1)/(x—1))/2

APENDICE I:

CONEXIONES DE DISPOSITIVOS DE ENTRADA/SALIDA

Este apéndice esta pensado para mostrarle que conexiones se pueden hacer con el Commodore 64.

1) Game I/O
2) Cartridge Slot
3) Audio/Video
4) Serial I/O (Disk/Printer)
5) Modulador Output
6) Cassete
7) User Port

Control del Puerto 1

Contacto Pua n.°	TIPO	N O T A
1	JOY A0	
2	JOY A1	
3	JOY A2	
4	JOY A3	
5	Potenciόm. A4	
6	botón A láp. óp.	
7	+5V	Máx. 50 mA
8	TIERRA	
9	Potenciόm. AX	

1 2 3 4 5
6 7 8 9

Control del Puerto 2

Contacto Pua n.°	TIPO	N O T A
1	JOY B0	
2	JOY B1	
3	JOY B2	
4	JOY B3	
5	Potenciόm. B4	
6	botón B láp. óp.	
7	+5V	Máx. 50 mA
8	TIERRA	
9	Potenciόm. BX	

Expansión para cartuchos:

Contacto	Tipo
12	BA
13	DMA
14	D7
15	D6
16	D5
17	D4
18	D3
19	D2
20	D1
21	D0
22	TIERRA

Contacto	Tipo
1	TIERRA
2	+5V
3	+5V
4	IRQ
5	R/W
6	Dot Clock
7	I/0 1
8	GAME
9	EXROM
10	I/0 2
11	ROML

Contacto	Tipo
N	A9
P	A8
R	A7
S	A6
T	A5
U	A4
V	A3
W	A2
X	A1
Y	A0
Z	TIERRA

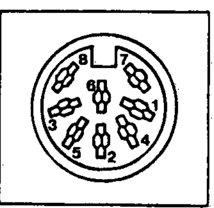
Contacto	Tipo
A	TIERRA
B	ROMH
C	RESET
D	NMI
E	S 02
F	A 15
G	A 14
H	13
I	A 12
J	A 11
K	A 10

22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1

Z Y X W V U T S R P N M L K J H F E D C B A

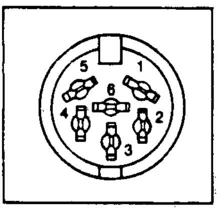
Audio/Video

Contacto	Tipo	Nota
1	LUM/SYNC	Salida Lum. SYNC.
2	GND	
3	AUDIO OUT	
4	VIDEO OUT	Salida Video Comp
5	AUDIO IN	
6	COLOR OUT	Salida Cromo
7	NC	NC
8	NC	NC



E/S de Serie (impresora/disco)

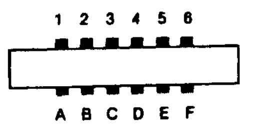
Contacto	Tipo
1	SERIAL SQRIN
2	TIERRA
3	SERIAL ANT. IN/OUT
4	SERIAL CLK IN/OUT
5	SERIAL DATA IN/OUT
6	RESET





El Cassette

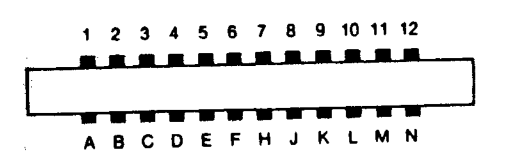
Contacto	Tipo
A-1	TIERRA
B-2	+5V
C-3	MOTOR DEL CASSETTE
D-4	LECTURA DEL CASSETTE
E-5	GRABACION DEL CASSETTE
F-6	INTERRUPTOR CASSETTE



El Puerto (Port) del Usuario:

Contacto	Tipo	Nota
1	TIERRA	
2	+5V	Máx. 100 mA
3	RESET	
4	CNT1	
5	SP1	
6	C N T 2	
7	S P 2	
8	P C 2	
9	ENT. SERIAL ANT.	
10	9V≃	Máx. 100 mA
11	9V≃	Máx. 100 mA
12	TIERRA	

Contacto	Tipo	Nota
A	TIERRA	
B	FLAG2	
C	PB0	
D	PB1	
E	PB2	
F	PB3	
H	PB4	
J	PB5	
K	PB6	
L	PB7	
M	PA2	
N	TIERRA	



APENDICE J:

PROGRAMAS A PROBAR

Hemos incluido (en este manual) un gran número de programas útiles para que los pruebe con su Commodore 64. Estos programas son divertidos y útiles.


```
100 PRINT"♥ESCRITURA        JIM BUTTERFIELD"
120 INPUT"♥QUIERE INSTRUCCIONES";Z$:IFASC(Z$)=78GOTO250
130 PRINT"♥INTENTE ADIVINAR LA PALABRA MISTERIOSA DE 5 LETRAS"
140 PRINT"♥DEBE ADIVINAR SOLO 5 LETRAS CORRECTAS"
150 PRINT"DE LA PALABRA,ADEMAS..."
160 PRINT"DEBE CONTAR EL NUMERO DE PARTIDAS"
170 PRINT"(O 'INTENTOS') EN SU PARTIDA."
180 PRINT"♥SUGERENCIA:LA SUERTE VARIA LEVEMENTE"
190 PRINT"  DE UN INTENTO AL SIGUIENTE; ASI"
200 PRINT"  SI ESCRIBE 'BARCO' Y LOGRA 2 PUNTOS"
210 PRINT"  PUEDE INTENTAR 'BARCA' O 'MARCO'"
220 PRINT"  EN EL PROXIMO INTENTO..."
250 DATABXBSF,IPCCZ,DBDIF,ESFBE,PGGBM
260 DATAHPSHF,IBUDI,DJWJM,KPMMZ,IBZBL
270 DATASBKBI,MFWFM,NJNJD,BOOFY,QJQFS
280 DATARVFTU,SJWFS,QSFTT,PUUFS,FWFOU
290 DATAXFBWF,FYUPM,NVTIZ,AFCSB,GJAAZ
300 DATAUIJOL,ESVOL,GMPFE,UJHFS,GBLFS
310 DATACPPUJ,MZJOH,TRVBU,HBVAF,PXJOH
320 DATAUISFF,TJHIU,BYMFT,HSVN0,BSFOB
330 DATARVBSU,DSFFQ,CFMOI,QSFTT,T0BSL
340 DATASBEBS,SVSBM,TNFMM,GSPXO,ESJGU
400 N=50
410 DIMN$(N),Z(5),Y(5)
420 FORJ=1TON:READN$(J):NEXTJ
430 T=TI
440 T=T/1000:IFT>=1THEN440
450 Z=RND(-T)
460 G=0:N$=N$(RND(1)*N+1)
500 G=0:N$=N$(RND(1)*N+1)
510 PRINT"♥TIENE UNA PALABRA DE 5 LETRAS:":IFR>0GOTO560
520 PRINT"PRUEBE (CON PALABRAS CORRECTAS)"
530 PRINT"Y CUENTE CUANTAS"
540 PRINT"'PAREJAS', O LETRAS EMPAREJADAS,"
550 PRINT"TIENE...."
560 G=G+1:INPUT"SU PALABRA";Z$
570 IFLEN(Z$)<>5THENPRINT"PUEDE PROBAR UN PALBRA DE 5
580 V=0:H=0:M=0                                 LETRAS!":GOTO560
590 FORJ=1TO5
600 Z=ASC(MID$(Z$,J,1)):Y=ASC(MID$(N$,J,1))-1:IFY=64THENY=90
610 IFZ<65ORZ>90THENPRINT"NO ES UNA PALABRA!":GOTO560
620 IFZ=65ORZ=69ORZ=73ORZ=79ORZ=85ORZ=89THENV=V+1
630 IFZ=YTHENM=M+1
640 Z(J)=Z:Y(J)=Y:NEXTJ
650 IFM=5GOTO800
660 IFV=0ORV=5THENPRINT"DIGAME..DE QUE CLASE ES ESTA
670 FORJ=1TO5:Y=Y(J)                          PALABRA?":GOTO560
680 FORK=1TO5:IFY=Z(K)THENH=H+1:Z(K)=0:GOTO700
690 NEXTK
700 NEXTJ
710 PRINT"]▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮";H;"PUNTOS"
720 IFG<30GOTO560
730 PRINT"MEJORESE.. SON PALABRAS ";
740 FORJ=1TO5:PRINTCHR$(Y(J));:NEXTJ
750 PRINT"'":GOTO810
800 PRINT"TIENE SOLO";G;"INTENTOS."
810 PRINT"♥OTRA PALABRA";Z$
820 R=1:IFASC(Z$)<>78GOTO500
READY.
```




```
1 REM *** SECUENCIA
2 REM
3 REM *** DE PET USER GROUP
4 REM *** SOFTWARE EXCHANGE
5 REM *** PO BOX 371
6 REM *** MONTGOMERYVILLE, PA 18936
7 REM
50 DIMA$(26)
100 Z$="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
110 Z1$="12345678901234567890123456"
200 PRINT"■■■ENTRE LA CADENA EN UNA SECUENCIA■"
220 INPUT"DE UNA LONGITUD MAXIMA DE 26 ";S%
230 IFS%<1ORS%>26THEN200
240 S=S%
300 FOR I=1TOS
310 A$(I)=MID$(Z$,I,1)
320 NEXTI
400 REM CADENA AL AZAR
420 FORI=1TOS
430 K=INT(RND(1)*S+1)
440 T$=A$(I)
450 A$(I)=A$(K)
460 A$(K)=T$
470 NEXTI
480 GOSUB950
595 T=0
600 REM RESERVA SUBCADENA
605 T=T+1
610 INPUT"CUANTOS INVIERTO ";R%
620 IFR%=0GOTO900
630 IFR%>0ANDR%<=SGOTO650
640 PRINT"PUEDE SER ENTRE 1 Y ";S:GOTO610
650 R=INT(R%/2)
660 FORI=1TOR
670 T$=A$(I)
680 A$(I)=A$(R%-I+1)
690 A$(R%-I+1)=T$
700 NEXTI
750 GOSUB950
800 C=1:FORI=2TOS
810 IFA$(I)>A$(I-1)GOTO830
820 C=0
830 NEXTI
840 IFC=0GOTO600
850 PRINT"■HA HECHO ";T;" INTENTOS"
900 REM PRUEBA PARA OTRO JUEGO
910 INPUT"■QUIERE JUGAR OTRA ";Y$
920 IFLEFT$(Y$,1)="S"ORY$="OK"ORY$="1"GOTO200
930 END
950 PRINT
960 PRINTLEFT$(Z1$,S)
970 FORI=1TOS:PRINTA$(I);:NEXTI
980 PRINT"■"
990 RETURN
READY.
```



```
READY.

 90 REM TECLADO DE PIANO
 100 PRINT"■ ■ ■ ■  | ■ ■ ■  | ■ ■  | ■ ■ ":REM"■ ES SHIFT
     CLR/HOME; ■ ES CTRL-9
 110 PRINT" ■ ■ ■  | ■ ■ ■  | ■ ■  | ■ ■ " :REM" | ES SHIFT Y"
 120 PRINT" ■ ■ ■  | ■ ■ ■  | ■ ■  | ■ ■ "
 130 PRINT" ■ | | | | | | | | | | | | "
 140 PRINT" ■Q W E R T Y U I O P @ * ↑ |"
 150 PRINT"■'ESPACIO' PARA SOLO O POLIFONICO":REM"■
     ES CRSR ABAJO"
 160 PRINT"■'F1,F3,F5,F7' SELECCION OCTAVA"
 170 PRINT"■'F2,F4,F6,F8' FORMA DE ONDA■"
 180 PRINT"PARA PENDIENTE, CONSULTE LA TABLA DE
     FRECUENCIAS..."
 190 S=13*4096+1024:DIMF(26):DIMK(255)
 200 FORI=0TO28:POKES+I,0:NEXT
 210 F1=7040:FORI=1TO26:F(27-I)=F1*5.8+30:F1=F1/2↑(1/12):NEXT
 220 K$="Q2W3ER5T6Y7UI9O0P@+*-↑"
 230 FORI=1TOLEN(K$):K(ASC(MID$(K$,I)))=I:NEXT
 240 PRINT"■                              ":REM"■ ES SHIFT CRSR ABAJO
 250 AT=0:DE=0:SU=15:RE=9:SV=SU*16+RE:AV=AT*16+DE:WV=16:W=0:
     M=1:OC=4:HB=256:Z=0
 260 FORI=0TO2:POKES+5+I*7,AT*16+DE:POKES+6+I*7,SU*16+RE
 270 POKES+2+I*7,4000AND255:POKES+3+I*7,4000/256:NEXT
 280 POKES+24,15:REM+16+64:POKES+23,7
 300 GETA$:IFA$=""THEN300
 310 FR=F(K(ASC(A$)))/M:T=V*7:CR=S+T+4:IFFR=ZTHEN500
 320 POKES+6+T,Z:REM FINAL DEC/SOS
 325 POKES+5+T,Z:REM FINAL ATA/EMI
 330 POKECR,8:POKECR,0:REM FIX FUERA
 340 POKES+T,FR-HB*INT(FR/HB):REM PONE BAJO
 350 POKES+1+T,FR/HB:REM PONE ALTO
 360 POKES+6+T,SV:REM PONE DEC/SOS
 365 POKES+5+T,AV:REM PONE ATA/EMI
 370 POKECR,WV+1:FORI=1TO50*AT:NEXT
 375 POKECR,WV:REM PULSO
 380 IFP=1THENV=V+1:IFV=3THENV=0
 400 GOTO300
 500 IFA$="■"THENM=1:OC=4:GOTO300:REM" ES F1
 510 IFA$="■"THENM=2:OC=3:GOTO300:REM" ES F3
 520 IFA$="■"THENM=4:OC=2:GOTO300:REM" ES F5
 530 IFA$="■"THENM=8:OC=1:GOTO300:REM" ES F7
 540 IFA$="■"THENW=0:WV=16:GOTO300:REM"ES SHIFT F1
 550 IFA$="■"THENW=1:WV=32:GOTO300:REM" ES SHIFT F3
 560 IFA$="■"THENW=2:WV=64:GOTO300:REM" ES SHIFT F5
 570 IFA$="■"THENW=3:WV=128:GOTO300:REM" ES SHIFT F7
 580 IFA$=" "THENP=1-P:GOTO300
 590 IFA$="■"THEN200
 600 GOTO300
 800 PRINT"PULSE UNA TECLA"
 810 GETA$:IFA$=""THEN810:REM ESPERA UNA TECLA
 820 PRINTA$:RETURN
READY.
```

APENDICE K

CONVERSION DE PROGRAMAS BASIC ESTANDARD AL (BASIC DEL) COMMODORE 64

Si tiene programas escritos en otro BASIC que el de Commodore, será necesario hacer unos pequeños retoques, antes de lanzarlos con el Commodore 64, hemos incluido algunos consejos para hacer la conversión más sencilla.

Dimensiones de Cadenas

Borre todas las instrucciones que declaran la longitud de la cadena. Una instrucción DIM A$(I,J) que dimensiona una matriz de cadenas para J elementos de longitud I, deberan ser convertidos a la sentencia de BASIC Commodore DIM A$ (J)

Algunos BASICs utilizan una coma o signos para la concatenación de cadenas. Cada una de ellas debe ser cambiada por un signo " + " que es el operador del BASIC-Commodore para la concatación de cadenas (alfabéticas).

En el Basic del Commodore 64, las funciones MID$, RIGHT$: LEFT$, se utilizan para obtener subcadenas de cadenas. Formatos como A$(1) para acceder al carácter de orden I en A$, o A$(I,J) para sacar una subcadena de A$ de la posición I a J, deben cambiarse así:

Otros BASICs	BASIC del Commodore 64
A$(I) = X$	A$ = LEFT$(A$,I—1) + X$ + MID$(A$,I + 1)
A$(I,J) = X$	A$ = LEFT$(A$,I—1) + X$ + MID$(A$,J + 1)

Asignaciones Múltiples

Para poner a cero. B y C algunos BASICs le permiten instrucciones de este tipo:

```
1Ø LET B = C = Ø
```


El BASIC del Commodore 64 interpretaría el segundo signo " = " como un operador lógico y ajustaría B = —1 si C = 0. Para evitar este problema convierte la instrucción en la siguiente:

```
1Ø C = Ø : B = Ø
```

Instrucciones Múltiples

Algunos BASICs utilizan una barra (atras) " " para separar diversas instrucciones en una misma línea. Con el BASIC-Commodore separe todas las instrucciones con los 2 puntos (:)

Funciones MAT

Los programas que utilizan las funciones MAT algunos BASICs deben ser escritas utilizando un bucle FOR...NEXT para ser ejecutadas debidamente.



APENDICE L:

MENSAJES DE ERROR

Este Apéndice contiene una lista completa de los mensajes de error generados por el Commodore 64 y la descripción de las causas:

BAD DATA...(Datos incorrectos): Se recibieron datos alfanuméricos desde un fichero abierto, cuando el programa esperaba datos numéricos.

BAD SUBSCRIPT...(Subscrito incorrecto): El programa intentaba hacer referencia a un elemento de una matriz cuyo número estaba fuera de la gama especificada en la instrucción DIM.

CAN'T CONTINUE...(No es posible continuar): El comando CONT no puede operar porque no había empezado la ejecución del programa, hay un error o se ha cambiado una línea.

DEVICE NOT PRESENT...(Falta el periférico): el periférico requerido no está disponible para una instrucción OPEN, CLOSE, CMD, PRINT#, INPUT# o GET#

DIVISION BY ZERO...(División por cero): La división por cero no tiene sentido matemático y no está permitida.

EXTRA IGNORED...(Se omiten los datos extra): Se mecanografiaron demasiados datos en respuesta a una instrucción INPUT. Se aceptaron sólo los primeros datos.

FILE NOT FOUND...(Fichero no encontrado): Si usted buscaba un fichero en una cinta, se encontró una marca END OF TAPE (final de cinta). Si se buscaba en un diskette, no existe fichero con dicho nombre.

FILE NOT OPEN...(Fichero no abierto): El fichero especificado en una instrucción CLOSE, CMD, PRINT#, INPUT# o GET# debe abrirse primero con OPEN.

FILE OPEN...(Fichero abierto): Se ha intentado abrir un fichero usando el número de un fichero ya abierto.

FORMULA TOO COMPLEX...(Fórmula demasiado completa): La expresión de cadena evaluada debería dividirse por lo menos en dos partes para que el sistema pudiera elaborarla.

ILLEGAL DIRECT...(Ilegal directo): La instrucción INPUT sólo puede usarse incorporada en un programa, no en modo directo.

ILLEGAL QUANTITY...(Cantidad ilegal): Un número usado como el argumento de una función o instrucción está fuera de la gama permitida.

LOAD: Hay un problema con el programa en la cinta.



NEXT WITHOUT FOR...(NEXT sin FOR): Hay bucles encajados incorrectamente o un nombre de variable en una instrucción NEXT que no corresponde a la de la instrucción FOR.

NOT INPUT FILE...(No fichero entrada): Se ha intentado introducir o extraer datos con INPUT o GET en un fichero especificado sólo para salida de datos.

NOT OUTPUT FILE...(No fichero de salida): Se ha intentado grabar datos con PRINT# en un fichero que estaba especificado sólo para extracción de datos.

OUT OF DATA...(Datos agotados): Se ejecutó una instrucción READ, pero no quedaban datos por leer en la instrucción DATA.

OUT OF MEMORY...(Memoria agotada): No queda memoria RAM para el programa o sus variables. Puede ocurrir también cuando se encajan demasiados bucles FOR o hay demasiadas instrucciones GOSUB.

OVERFLOW...(No cabe): El resultado de un cálculo es mayor que el máximo número permitido, que es 1.70141884/ + 38.

REDIM'D ARRAY...(Matriz redimensionada): Una matriz puede dimensionarse sólo una vez, con DIM. Si se usa una variable de matriz antes de que ésta sea dimensionada, se ejecuta una operación DIM automáticamente, estableciendo el número de elementos de dicha matriz en diez, y cualquier posterior dimensionado generará el mensaje de error.

RED FROM START...(Volver a empezar desde el principio): Se mecanografiaron datos de caracteres durante una instrucción INPUT, en vez de los datos numéricos que se esperaban. Simplemente volver a efectuar la entrada en el teclado en modo que sea correcta y el programa continuará por sí mismo.

RETURN WITHOUT GOSUB...(RETURN sin GOSUB): Se encontró una instrucción RETURN sin que se hubiera incorporado un comando GOSUB.

STRING TOO LONG...(Cadena demasiado larga): Una cadena puede contener como máximo 255 caracteres.

SYNTAX...(Sintaxis): Una instrucción no es reconocida por el VIC: hay un paréntesis de más o de menos, un error de mecanografiado, etc.

TYPE MISMATCH...(Los datos mecanografiados no coinciden): Este mensaje de error se genera cuando se usa un número en vez de una cadena, o viceversa.

UNDEF'D FUNCTION...(Función no definida): Se usó como referencia una función definida por el usuario, pero la misma no había sido definida usando la instrucción DEF FN.

UNDEF'D STATEMENT...(Instrucción no definida): Se intentó pasar con GOTO o GOSUB o ejecutar con RUN una línea que no existía.

VERIFY...(Verificar): El programa en la cinta cassette o en el disco no coincide con el programa actualmente en la memoria del ordenador.



APENDICE M:

VALORES DE LAS NOTAS MUSICALES

Este Apéndice contiene una lista completa de Notas musicales con el número de nota, el nombre de la nota, y los códigos a entrar por POKE en los registros de Alta y Baja Frecuencia del chip de sonidos para generar la nota deseada.

N.º de Nota	Nota-Octava	Alta Frecuencia	Baja Frecuencia
0	C–0	1	18
1	C#–0	1	35
2	D–0	1	52
3	D#–0	1	70
4	E–0	1	90
5	F–0	1	110
6	F#–0	1	132
7	G–0	1	155
8	G#–0	1	179
9	A–0	1	205
10	A#–0	1	233
11	B–0	2	6
12	C–1	2	37
13	C#–1	2	69
14	D–1	2	104
15	D#–1	2	140
16	E–1	2	179
17	F–1	2	220
18	F#–1	3	8
19	G–1	3	54
20	G#–1	3	103
21	A–1	3	155
22	A#–1	3	210
23	B–1	4	12
24	C–2	4	73
25	C#–2	4	139
26	D–2	4	208
27	D#–2	5	25
28	E–2	5	103
29	F–2	5	185
30	F#–2	6	16
31	G–2	6	108
32	G#–2	6	206

N.º de Nota	Nota-Octava	Alta Frecuencia	Baja Frecuencia
33	A–2	7	53
34	A#–2	7	163
35	B–2	8	23
36	C–3	8	147
37	C#–3	9	21
38	D–3	9	159
39	D#–3	10	60
40	E–3	10	205
41	F–3	11	114
42	F#–3	12	32
43	G–3	12	216
44	G#–3	13	156
45	A–3	14	107
46	A#–3	15	70
47	B–3	16	47
48	C–4	17	37
49	C#–4	18	42
50	D–4	19	63
51	D#–4	20	100
52	E–4	21	154
53	F–4	22	227
54	F#–4	24	63
55	G–4	25	177
56	G#–4	27	56
57	A–4	28	214
58	A#–4	30	141
59	B–4	32	94
60	C–5	34	75
61	C#–5	36	85
62	D–5	38	126
63	D#–5	40	200
64	E–5	43	52
65	F–5	45	198
66	F#–5	48	127
67	G–5	51	97
68	G#–5	54	111
69	A–5	57	172
70	A#–5	61	126
71	B–5	64	188
72	C–6	68	149
73	C#–6	72	169
74	D–6	76	252
75	D#–6	81	161
76	E–6	86	105
77	F6	91	140
78	F#–6	96	254
79	G–6	102	194
80	G#–6	108	223

N.º de Nota	Nota-Octava	Alta Frecuencia	Baja Frecuencia
81	A–6	115	88
82	A#–6	122	52
83	B–6	129	120
84	C–7	137	43
85	C#–7	145	83
86	D–7	153	247
87	D#–7	163	31
88	E–7	172	210
89	F–7	183	25
90	F#–7	193	252
91	G–7	205	133
92	G#–7	217	189
93	A–7	230	176
94	A#–7	244	103



APENDICE N:

BIBLIOGRAFIA

Addison-Wesley — "BASIC and the Personal Computer", Dwyer and Critchfield

Compute — "Compute's First Book of PET/CBM"

Cowbay Computing — "Feed Me, I'm Your PET Computer", Carol Alexander

"Looking Good with Your PET", Carol Alexander

"Teacher's PET — Plans, Quizzes, and Answers"

Creative Computing — "Getting Acquainted With Your VIC 20", T. Hartnell

Dilithium Press — "BASIC Basic-English Dictionary for the PET", Larry Noonan

"PET BASIC", Tom Rugg and Phil Feldman

Faulk Baker Associates — "MOS Programming Manual", MOS Technology

Hayden Book Co. — "BASIC From the Ground Up", David E. Simon

"I Speak BASIC to My PET", Aubrey Jones, Jr.

"Library of PET Subroutines", Nick Hampshire

"PET Graphics", Nick Hampshire

"BASIC Conversions Handbook, Apple, TRS-80, and PET", David A. Brain, Phillip R. Oviatt, Paul J. Paquin, and Chandler P. Stone

Howard W. Sams — "The Howard W. Sams Crash Course in Microcomputers", Louis E. Frenzel, Jr.

"Mostly BASIC: Applications for Your PET", Howard Berenbon

"PET Interfacing", James M. Downey and Steven M. Rogers

"VIC 20 Programmer's Reference Guide", A. Finkel, P. Higginbottom, N. Harris, and M. Tomczy.

Little, Brown & Co. — "Computer Games for Businesses, Schools, and Homes", J. Victor Nagigian, and William S. Hodges

"The Computer Tutor: Learning Activities for Homes and Schools", Gary W. Orwig, University of Central Florida, and William S. Hodges.



McGraw-Hill

"Hands-On BASIC With a PET". Herbert D. Peckman

"Home and Office Use of VisiCalc". D. Castlewitz, and L. Chisauki.

Osborne/McGraw-Hill

Guía del Ordenador Personal PET/CBM. Carroll S. Donahue

"PET Fun and Games", R. Jeffries and G. Fisher

"PET and the IEEE". A. Osborne and C. Donahue

"Some Common BASIC Programs for the PET". L. Poole, M. Borchers, and C. Donahue

"Sistema operativo CP/M. Guia del Usuario". Thom Hogan

"CBM Professional Computer Guide"

"The PET Personal Guide"

"The 8086 Book". Russell Rector and George Alexy

P.C. Publications

"Beginning Self-Teaching Computer Lessons"

Prentice-Hall

"The PET Personal Computer for Beginners", S. Dunn and V. Morgan

Reston Publishing Co.

"PET and the IEEE 488 Bus (GPIB)", Eugene Fisher and C. W. Jensen

"PET BASIC — Training Your PET Computer", Ramon Zamora, Wm. F. Carrie. and B. Allbrecht

"PET Games and Recreation", M. Ogelsby, L. Lindsey, and D. Kunkin

"PET BASIC", Richard Huskell

"VIC Games and Recreation"

Telmas Courseware Ratings

"BASIC and the Personal Computer", T.A. Dwyer, and M. Critchfield

Total Information Services

"Understanding Your PET/CBM, Vol. 1, BASIC Programming"

"Understanding Your VIC", David Schultz

APENDICE O:

MAPA DE LOS REGISTROS DE SPRITES

Registro N.º Decim.	Hexag.	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
0	0	S0X7							S0X0	Componente X SPRITE 0
1	1	S0Y7							S0Y0	Componente Y SPRITE 0
2	2	S1X7							S1X0	SPRITE 1X
3	3	S1Y7							S1Y0	SPRITE 1Y
4	4	S2X7							S2X0	SPRITE 2X
5	5	S2Y7							S2Y0	SPRITE 2Y
6	6	S3X7							S3X0	SPRITE 3X
7	7	S3Y7							S3Y0	SPRITE 3Y
8	8	S4X7							S4X0	SPRITE 4X
9	9	S4Y7							S4Y0	SPRITE 4Y
10	A	S5X7							S5X0	SPRITE 5X
11	B	S5Y7							S5Y0	SPRITE 5Y
12	C	S6X7							S6X0	SPRITE 6X
13	D	S6Y7							S6Y0	SPRITE 6Y
14	E	S7X7							S7X0	Componente X SPRITE 7
15	F	S7Y7							S7Y0	Componente Y SPRITE 7
16	10	S7X8	S6X8	S5X8	S4X8	S3X8	S2X8	S1X8	S0X8	MSB de Coordenad. X
17	11	RC8	EC5	BSM	BLNK	RSEL	YSCL2	YSCL1	YSCL0	Scroll/Modo
18	12	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0	RASTREO
19	13	LPX7							LPX0	LAPIZ-OPT. X
20	14	LPY7							LPY0	LAPIZ-OPT. Y

Registro N.º Decim.	Hexag.	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
21	15	SE7							SE0	LANZAMIEN. SPRITE ON/OFF
22	16	N.C.	N.C.	RST	MCM	CSEL	XSCL2	XSCL1	XSCL0	Scroll/Modo
23	17	SEXY7							SEXY0	ANULACION SPRITE Y
24	18	VS13	VS12	VS11	CB13	CB12	CB11	CB10	N.C.	Memoria de caracter de PANTALLA
25	19	IRQ	N.C.	N.C.	N.C.	LPIRQ	ISSC	ISBC	RIRQ	Interruptor DEMANDA
26	1A	N.C.	N.C.	N.C.	N.C.	MLPI	MISSC	MISBC	MRIRQ	Interruptor DEMANDA MASCARAS?
27	1B	BSP7							BSP0	Prioridad fondo SPRITE
28	1C	SCM7							SCM0	SELECTOR COLOR SPRITE
29	1D	SEXX7							SEXX0	AMPLIACION SPRITE X
30	1E	SSC7							SSC0	CHOQUE DE SPRITES
31	1F	SBC7							SBC0	CHOQUE DE FONDO Y SPRITE



		CODIGOS DE COLOR				DEC.	HEXA.		COLOR
32	20	0	0	NEGRO	EXT 1				COLOR EXTERIOR
33	21	1	1	BLANCO	BKGD0				
34	22	2	2	ROJO	BKGD1				
35	23	3	3	TURQUESA	BKGD2				
36	24	4	4	MALVA/VIOLETA	BKGD3				
37	25	5	5	VERDE	SMC 0				SPRITE MULTICOLOR 0
38	26	6	6	AZUL	SMC 1				1
39	27	7	7	AMARILLO	S0COL				SPRITE COLOR 0
40	28	8	8	NARANJA	S1COL				1
41	29	9	9	MARRON	S2COL				2
42	2A	10	A	ROJO	S3COL				3
43	2B	11	B	GRIS 1	S4COL				4
44	2C	12	C	GRIS 2	S5COL				5
45	2D	13	D	VERDE CL.	S6COL				6
46	2E	14	E	AZUL CL.	S7COL				7
		15	F	GRIS 3					

LEYENDA:
SOLO LOS COLORES DE 0 A 7 PUEDEN SER UTILIZADOS EN MULTICOLOR.



APENDICE P:

BASES DEL CONTROL DE SONIDO DEL COMMODORE 64

Esta tabla contiene los códigos numéricos que necesita usar en sus programas de sonido, de acuerdo con las tres voces del Commodore 64 que quiera usar. Para situar y ajustar el control de sonido en su programa BASIC, POKE el número de la segunda columna, seguido de coma y un número de la tabla... semejante a este: POKE 54276,17 (selecciona la Forma de onda triangular para la VOZ 1).

Recuerde que debe ajustar el VOLUMEN antes de que pueda generar sonido: POKE 54296 seguido de un número de 0 a 15 ajusta el volumen para las 3 voces.

Tome 2 POKEs separados para generar cada nota musical... por ejemplo POKE 54273,34: POKE 54272,75 designa C baja en la escala vocal sencilla.

También... no ha limitado los números mostrados en las tablas. Si 34 no es un sonido "correcto" para C baja, pruebe 35. Coloque un SOSTEN alto o ATAQUE proporcional que estos muestran, más dos o más números SOSTEN unidos. (Ejemplo: POKE54277,96 combina dos ataques proporcionales... pero... POKE 54277,20 provee un ataque bajo proporcional (16) y una caida media proporcional (4).

AJUSTE VOLUMEN — MISMO PARA LAS 3 VOCES

CONTROL VOLUM.	POKE54296	Settings range from 0 (off) to 15 (loudest)

VOZ NUMERO 1

PARA CONTROLAR	ESTE N.° POKE	SEGUIDO DE UNO DE ESTOS NUMEROS (0 a 15 ... o ... 0 a 255 depende del rango)

TOCA UNA NOTA	C	C#	D	D#	E	F	F#	G	G#	A	A#	B	C	C#
ALTA FRECUEN.	54273 34	36	38	40	43	45	48	51	54	57	61	64	68	72
BAJA FRECUEN.	54272 75	85	126	200	52	198	127	97	111	172	126	188	149	169

FORMA DE ONDA	POKE	TRIANGULO	D. SIERRA	PULSO	RUIDO
	54276	17	33	65	129

PULSO PROPORCIONAL (Forma de onda)		
PULSO ALTO	54275	Un valor de 0 a 15 (para Pulso sólo)
PULSO BAJO	54274	Un valor de 0 a 255 (par Pulso sólo)

ATAQUE/CAIDA	POKE	AT4	AT3	AT2	AT1	CA4	CA3	CA2	CA1
	54277	128	64	32	16	8	4	2	1

SOSTEN/ESCAPE	POKE	SO4	SO3	SO2	SO1	ES4	ES3	ES2	ES1
	54278	128	64	32	16	8	4	2	1

VOZ NUMERO 2

TOCA UNA NOTA	C	C#	D	D#	E	F	F#	G	G#	A	A#	B	C	C#
ALTA FRECUEN.	54280 34	36	38	40	43	45	48	51	54	57	61	64	68	72
BAJA FRECUEN.	54279 75	85	126	200	52	198	127	97	111	172	126	188	149	169

FORMA DE ONDA	POKE	TRIANGULO	D. SIERRA	PULSO	RUIDO
	54288	17	33	65	129

PULSO PROPORCIONAL		
PULSO ALTO	54282	Un valor de 0 a 15 (para Pulso sólo)
PULSO BAJO	54281	Un valor de 0 a 255 (par Pulso sólo)

ATAQUE/CAIDA	POKE	AT4	AT3	AT2	AT1	CA4	CA3	CA2	CA1
	54284	128	64	32	16	8	4	2	1

SOSTEN/ESCAPE	POKE	SO4	SO3	SO2	SO1	ES4	ES3	ES2	ES1
	54286	128	64	32	16	8	4	2	1



VOZ NUMERO 3

TOCA UNA NOTA	C	C#	D	D#	E	F	F#	G	G#	A	A#	B	C	C#
ALTA FRECUEN.	54287 34	36	38	40	43	45	48	51	54	57	61	64	68	72
BAJA FRECUEN.	54286 75	85	126	200	52	198	127	97	111	172	126	188	149	169

FORMA DE ONDA	POKE	TRIANGULO	D. SIERRA	PULSO	RUIDO
	54290	17	33	65	129

PULSO PROPORCIONAL

PULSO ALTO	54289	Un valor de 0 a 15 (para Pulso sólo)
PULSO BAJO	54288	Un valor de 0 a 255 (par Pulso sólo)

ATAQUE/CAIDA	POKE	AT4	AT3	AT2	AT1	CA4	CA3	CA2	CA1
	54291	128	64	32	16	8	4	2	1

SOSTEN/ESCAPE	POKE	SO4	SO3	SO2	SO1	ES4	ES3	ES2	ES1
	54292	128	64	32	16	8	4	2	1

PRUEBE ESTOS AJUSTES PARA SIMULAR DIFERENTES AJUSTES

Instrumento	Form. onda	A/C	S/E	Pro. Pulso
Piano	Pulso	9	0	Alto-0,Bajo-255
Flauta	Triángulo	96	0	No aplicable
Clavicordio	D. Sierra	9	0	No aplicable
Xilofono	Triángulo	9	0	No aplicable
Organo	Triángulo	0	240	No aplicable
Colliape	Triángulo	0	240	No aplicable
Acordeon	Triángulo	102	0	No aplicable
Trompeta	D. Sierra	96	0	No aplicable

SIGNIFICADOS DE LOS TERMINOS SONOROS

ADSR—Ataque/Caida/Sostén/Escape
Ataque—Relación de elevación de sonido al volumen de pico
Caida—Relación de caida desde el volumen de Pico hasta el nivel sostén.
Sostén—Prolonga la nota a un cierto volumen
Escape—Relación a un volumen bajo para un nivel Sostén
Forma Onda—"Forma" de la onda Sonora
Pulso—Calidad del Tono en la Forma de onda Pulso

NOTA: Ataque/Caida y Sostén/Escape podría ajustarse siempre son POKEs en su programa ANTES que POKE la Forma de onda.

INDICE


microelectronica y control s a




microelectronica y control s a

GUIA RAPIDA DE LAS INSTRUCCIONES [illegible]

VARIABLES SIMPLES

Tipo	*Nombre*	*Capacidad*
Real	XY	±1.70141183 E+38
		±2.93873588 E—39
Entero	XY%	±32767
de Cadena	XY$	0 a 255 caracteres

X es una letra de (A a Z) Y es una letra o número los nombres de las variables pueden exceder 2 caracteres pero sólo reconocera estos dos primeros.

VARIABLES DE MATRICES

Tipo	*Nombre*
Uni-dimensional	XY(5)
Bi-dimensional	XY(5,5)
Tri-dimensional	XY(5,5,5)

Las matrices de hasta 11 elementos (0-10) pueden ser utilizadas cuando quiera. Pero las matrices de más de 11 elementos necesitan ser DIMensionadas.

OPERADORES ARITMETICOS/ALGEBRAICOS

=	Asigna valores a una variable
—	Signo negativo
↑	Exponenciación
*	Multiplicación
/	División
+	Adición
—	Sustracción

OPERADORES LOGICOS

=	Igual
<>	No igual o diferente
<	Inferior
>	Superior
<=	Inferior o igual
>=	Superior o igual
NOT	No lógico
AND	Y lógico
OR	O lógico

La expresión es igual a 1 si verdadera, a 0 si falsa.

COMANDOS DE SISTEMA

LOAD "NOMBRE"	Carga el programa de la cinta (Cassette)
SAVE "NOMBRE"	Guarda el programa en cassette
LOAD "NOMBRE",8	Carga el programa del disco
SAVE "NOMBRE",8	Guarda el programa en disco
VERIFY "NOMBRE"	Verifica que el programa se haya GUARDADO sin errores
RUN	Ejecuta un programa
RUN xxx	Ejecuta el programa a partir de la línea xxx
STOP	Para la ejecución
END	Termina la ejecución
CONT	Continúa con la ejecución del programa ahí donde se paró
PEEK(X)	Muestra el contenido de la situación de memoria X
POKE X,Y	Cambia el contenido de la situación de memoria X por el valor Y
SYS xxxxx	Salta para ejecutar un programa en lenguaje máquina a partir de xxxxx
WAIT X,Y,Z	El programa espera hasta que el contenido de X sea diferente de 0 el programa curva que X EOR(0)Z y AND(y)Y sean diferentes de cero
USR(X)	Pasa el valor X a una subrutina de lenguaje máquina

EDICION Y FORMATEACION DE COMANDOS

LIST	Junta todo el programa
LIST A-B	Lista de las líneas A a B
REM mensaje	Este es un mensaje de Comentario que aparece en el listado pero que no se ignora durante la ejecución
TAB(X)	Utilizado en una instrucción PRINT, crea un espacio de X casillas
SPC(X)	Imprime X blancas en la línea
POS(X)	Pone el cursor en la posición especificada por X.
CLR/HOME	Posiciona el cursor en la segunda superficie izquierda de la pantalla
SHIFT CLR5HOME	Limpia la pantalla y pone el cursor en posición "HOME" (esq. sup. izq.)
SHIFT INST/DEL	Inserta espacios en la situación en que se encuentre el cursor
INST/DEL	Borra caracteres a partir de la posición en que se encuentra el cursor.
CTRL	Cuando se utiliza con las teclas numéricas de color selecciona el color de texto. Además se puede utilizar dentro de una instrucción PRINT.
Teclas CRSR	Desplazan el cursor arriba, abajo a la derecha a izquierda en la pantalla
Tecla C=	Cuando se utiliza conjuntamente con SHIFT cambia de Mayúsculas a Minúsculas y Gráficos y viceversa. Cuando se utilizan con teclas numéricas de color presenta el segundo juego de colores de texto

MATRICES Y CADENAS

DIM A(X,Y,Z)	Ajunta la máxima suscripción para A; reserva espacios para cálculos (X+T·Y+T·Z+1)
LEN (X$)	Esta función da el número de caracteres de X$
STR$ (X)	Convierte el valor numérico de X en una cadena.
VAL (X$)	Esta función transforma A$ en un valor numérico a partir del caracter no numérico
CHR$	Convierte el código numérico ASCII en el caracter correspondiente
ASC("X")	Esta da el código ASCII del 1er. caracter de la variable X$.
LEFT$(A$,X)	Esta función de la cadena que contiene los caracteres X más a la izquierda de la variable X$.
RIGHT$(A$,S)	Esta función de los caracteres X más a la derecha de la variable X$.
MID$(A$,X,Y)	Da los caracteres Y de la variable A$ empezando por el caracter n.° X.

[illegible]

INPUT A$ o A	Muestra un "?" en la pantalla y le pide de entrar una cadena o valor numérico
INPUT "ABC";A	Muestra el mensaje ABC y le pide que entre un valor (con A$ una cadena)
GET A$ o A	Espera que el usuario le entre un valor de 1 caracter alfanumérico o un valor numérico con A; No necesita pulsar RETURN
DATA A,"B",C	Inicializa un fuego de datos o números que leera la instrucción READ.
READ A$ o A	Asigna el próximo valor del DATA a A$ o A
RESTORE	Pone a cero el "puntero" de lectura de datos, para empezar de nuevo
PRINT "A=";A	Escribe la cadena "A=" y el valor en A";" con "," se tabula al próximo campo; respuesta en la pantalla.

[illegible]

GOTO X	Se transforma la ejecución a la línea X.
IF A=3THEN 0	Si la situación en *IF* es verdad entonces (THEN) ejecute el final de la instrucción
FOR A=1 TO 10 STEP 2:NEXT	Ejecuta todas las instrucciones entre FOR y NEXT con A de 1 a 10 saltando de 2 en 2, a menos que no se especifique el STEP y entonces salta de 1 en 1
NEXT A	Define el final de un bucle. A es opcional
GOSUB 2000	Transfiere la ejecución del programa a la subrutina de la línea 2000
RETURN	Marca el final de una subrutina. Vuelve a la instrucción GOSUB más reciente
ON X GOTO A,B...	Conecta con la línea n.° X de la lista que sigue al GOTO. Si X=1 entonces conecta con la línea A
ON X GOSUB A,B...	Conecta con la subrutina de la línea n.° X de la lista que sigue al GOSUB. Si X=1 entonces conecta con la línea A